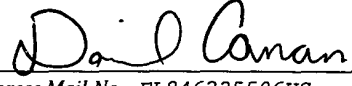


IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In Re U.S. Patent Application)
)
Applicant: Shibayama et al.)
)
Serial No.)
)
Filed: August 26, 2003)
)
For: PARALLEL PROCESS)
EXECUTION METHOD AND)
MULTIPROCESSOR)
COMPUTER)
)
Art Unit:)

I hereby certify that this paper is being deposited with the United States Postal Service as EXPRESS MAIL in an envelope addressed to: Mail Stop Patent Application, Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450, on this date.

8/26/2003
Date


Express Mail No. EL846225506US

CLAIM FOR PRIORITY

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Dear Sir:


Applicants claim priority benefits under 35 U.S.C. § 120 on the basis of the
PCT application identified below:

PCT Patent Application No. PCT/JP01/01532 filed February 28, 2001

A certified copy of the earlier application is enclosed.

Respectfully submitted,

GREER, BURNS & CRAIN, LTD.

By 

Patrick G. Burns
Registration No. 29,367

August 26, 2003
300 South Wacker Drive
Suite 2500
Chicago, Illinois 60606
Telephone: 312.360.0080
Facsimile: 312.360.9315

Patrick Burns
312-360-0080
11/28/01

日 本 国 特 許 庁

JAPAN PATENT OFFICE

別紙添付の書類は下記の出願書類の謄本に相違ないことを証明する。
This is to certify that the annexed is a true copy of the following application as filed
with this Office.

出 願 年 月 日
Date of Application: 2001年 2月28日

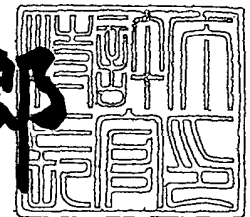
出 願 番 号
Application Number: PCT/JPO1/01532

出 願 人
Applicant (s): 富士通株式会社
柴山 悟基
松嶋 雄典
菊嶋 薫

2003 年 5 月 20 日

特許庁長官
Commissioner,
Japan Patent Office

太田 信一郎



出証平 15-500112

受理官庁用写し

1/5

特許協力条約に基づく国際出願願書

FUP-1042P

原本（出願用） - 印刷日時 2001年02月26日（26.02.2001）月曜日 14時40分32秒

0	受理官庁記入欄	
0-1	国際出願番号	PCT/JP01/01532
0-2	国際出願日	28.02.01
0-3	(受付印)	PCT International Application 日 本 国 特 許 庁
0-4	様式-PCT/RO/I01 この特許協力条約に基づく国際出願願書は、 0-4-1 右記によって作成された。	PCT-EASY Version 2.91 (updated 01.01.2001)
0-5	申立て 出願人は、この国際出願が特許協力条約に従って処理されることを請求する。	
0-6	出願人によって指定された受理官庁	日本国特許庁 (RO/JP)
0-7	出願人又は代理人の書類記号	FUP-1042P
I	発明の名称	並列プロセス実行方法、及びマルチプロセッサ型コンピュータ
II	出願人	
II-1	この欄に記載した者は	出願人である (applicant only)
II-2	右の指定国についての出願人である。	米国を除くすべての指定国 (all designated States except US)
II-4ja	名称	富士通株式会社
II-4en	Name	FUJITSU LIMITED
II-5ja	あて名:	211-8588 日本国 神奈川県 川崎市中原区上小田中 4 丁目 1 番 1 号
II-5en	Address:	1-1, Kamikodanaka 4-chome, Nakahara-ku Kawasaki-shi, Kanagawa 211-8588 Japan
II-6	国籍 (国名)	日本国 JP
II-7	住所 (国名)	日本国 JP
II-8	電話番号	044-754-3034
II-9	ファクシミリ番号	044-754-3563

特許協力条約に基づく国際出願願書

原本（出願用） - 印刷日時 2001年02月26日（26.02.2001）月曜日 14時40分32秒

FUP-1042P

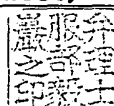
III-1 III-1-1	その他の出願人又は発明者 この欄に記載した者は	出願人及び発明者である (applicant and inventor) 米国のみ (US only)
III-1-2	右の指定国についての出願人である。	
III-1-4ja	氏名(姓名)	柴山 悟基
III-1-4en	Name (LAST, First)	SHIBAYAMA, Satoki
III-1-5ja	あて名:	211-8588 日本国 神奈川県 川崎市中原区上小田中 4 丁目 1 番 1 号
III-1-5en	Address:	富士通株式会社内 c/o FUJITSU LIMITED 1-1, Kamikodanaka 4-chome, Nakahara-ku Kawasaki-shi, Kanagawa 211-8588 Japan
III-1-6	国籍 (国名)	日本国 JP
III-1-7	住所 (国名)	日本国 JP
III-2 III-2-1	その他の出願人又は発明者 この欄に記載した者は	出願人及び発明者である (applicant and inventor) 米国のみ (US only)
III-2-2	右の指定国についての出願人である。	
III-2-4ja	氏名(姓名)	松嶋 雄典
III-2-4en	Name (LAST, First)	MATSUSHIMA, Yusuke
III-2-5ja	あて名:	211-8588 日本国 神奈川県 川崎市中原区上小田中 4 丁目 1 番 1 号
III-2-5en	Address:	富士通株式会社内 c/o FUJITSU LIMITED 1-1, Kamikodanaka 4-chome, Nakahara-ku Kawasaki-shi, Kanagawa 211-8588 Japan
III-2-6	国籍 (国名)	日本国 JP
III-2-7	住所 (国名)	日本国 JP

III-3 III-3-1	その他の出願人又は発明者 この欄に記載した者は	出願人及び発明者である (applicant and inventor) 米国のみ (US only)
III-3-2 III-3-4ja III-3-4en III-3-5ja	右の指定国についての出願人である。 氏名(姓名) Name (LAST, First) あて名:	菊嶋 薫 KIKUSHIMA, Kaoru 211-8588 日本国 神奈川県 川崎市中原区上小田中 4 丁目 1 番 1 号 富士通株式会社内 c/o FUJITSU LIMITED 1-1, Kamikodanaka 4-chome, Nakahara-ku Kawasaki-shi, Kanagawa 211-8588 Japan
III-3-5en	Address:	日本国 JP
III-3-6 III-3-7	国籍(国名) 住所(国名)	日本国 JP
IV-1 IV-1-1ja IV-1-1en IV-1-2ja	代理人又は共通の代表者、通知のあて名 下記の者は国際機関において右記のごとく出願人のために行動する。 氏名(姓名) Name (LAST, First) あて名:	代理人 (agent) 服部 毅巖 HATTORI, Kiyoshi 192-0082 日本国 東京都 八王子市東町 9 番 8 号 八王子東邦生命ビル 服部特許事務所 Hattori Patent Office Hachioji Tohoseimei Bldg. 9-8, Azuma-cho Hachioji-shi, Tokyo 192-0082 Japan
IV-1-2en	Address:	0426-45-6644 0426-45-8578
IV-1-3 IV-1-4	電話番号 ファクシミリ番号	
V V-1	国の指定 広域特許 (他の種類の保護又は取扱いを求める場合には括弧内に記載する。)	EP: AT BE CH&LI CY DE DK ES FI FR GB GR IE IT LU MC NL PT SE TR 及びヨーロッパ特許条約と特許協力条約の締約国である他の国
V-2	国内特許 (他の種類の保護又は取扱いを求める場合には括弧内に記載する。)	JP US

特許協力条約に基づく国際出願願書

FUP-1042P

原本（出願用） - 印刷日時 2001年02月26日（26.02.2001）月曜日 14時40分32秒

V-5	指定の確認の宣言 出願人は、上記の指定に加えて、規則4.9(b)の規定に基づき、特許協力条約のもとで認められる他の全ての国の指定を行う。ただし、V-6欄に示した国の指定を除く。出願人は、これらの追加される指定が確認を条件としていること、並びに優先日から15月が経過する前にその確認がなされない指定は、この期間の経過時に、出願人によって取り下げられたものとみなされることを宣言する。		
V-6	指定の確認から除かれる国	なし (NONE)	
VI	優先権主張	なし (NONE)	
VII-1	特定された国際調査機関 (ISA)	日本国特許庁 (ISA/JP)	
VIII	照合欄	用紙の枚数	添付された電子データ
VIII-1	願書	5	-
VIII-2	明細書	25	-
VIII-3	請求の範囲	4	-
VIII-4	要約	1	fup1042p.txt
VIII-5	図面	18	-
VIII-7	合計	53	
VIII-8	添付書類	添付	添付された電子データ
VIII-8	手数料計算用紙	✓	-
VIII-9	別個の記名押印された委任状		-
VIII-10	包括委任状の写し	✓	-
VIII-16	PCT-EASYディスク	-	フレキシブルディスク
VIII-17	その他	納付する手数料に相当する特許印紙を貼付した書面	-
VIII-17	その他	国際事務局の口座への振込を証明する書面	-
VIII-18	要約書とともに提示する図の番号	1	
VIII-19	国際出願の使用言語名:	日本語 (Japanese)	
IX-1	提出者の記名押印		
IX-1-1	氏名 (姓名)	服部 毅	

受理官庁記入欄

10-1	国際出願として提出された書類の実際の受理の日	28.02.01
10-2	図面:	
10-2-1	受理された	
10-2-2	不足図面がある	

特許協力条約に基づく国際出願願書

FUP-1042P

原本（出願用） - 印刷日時 2001年02月26日（26.02.2001）月曜日 14時40分32秒

10-3	国際出願として提出された書類を補完する書類又は図面であつてその後期間内に提出されたものの実際の受理の日（訂正日）	
10-4	特許協力条約第11条(2)に基づく必要な補完の期間内の受理の日	
10-5	出願人により特定された国際調査機関	ISA/JP
10-6	調査手数料未払いにつき、国際調査機関に調査用写しを送付していない	

国際事務局記入欄

11-1	記録原本の受理の日	
------	-----------	--

明 細 書

並列プロセス実行方法、及びマルチプロセッサ型コンピュータ

5 技術分野

本発明は並列プロセス実行方法、マルチプロセッサ型コンピュータ、並列プロセス実行プログラムおよびそのプログラムを記録した記録媒体に関し、特に並列プロセスと他のプロセスとを時分割で実行する並列プロセス実行方法、その方法を実行するマルチプロセッサ型コンピュータ、その方法をコンピュータに実行させる並列プロセス実行プログラムおよびそのプログラムを記録した記録媒体に関する。

背景技術

複数のプロセッサを搭載したコンピュータ（マルチプロセッサ型コンピュータ）では、1つのプログラムを複数のプロセッサで並列処理することが出来る。以下、並列処理可能なプログラムを並列プログラムと呼ぶ。並列プログラムを処理する場合、1つの並列プログラムから複数の並列プロセスが生成される。各並列プロセスは、互いに並行して実行可能なプロセスである。生成された複数の並列プロセスは、別々のプロセッサで並列に実行される。並列プロセスを実行する各プロセッサは互いにデータ交換をしながら、並列プログラムに定められた一連の処理を実行する。なお、ここでプロセスとは、1以上のスレッドを含む処理単位である。また、プロセッサとは、CPU(Central Processing Unit)やMPU(Micro Processing Unit)などの処理装置のことである。以下の説明では、便宜上、プロセッサとしてCPUを用いるものとする。

各並列プロセスの処理には、他の並列プロセスとの間でデータ交換（同期通信）を行うべきチェックポイントがある。データ交換が必要な2つの並列プロセスを実行する各CPUは、並列プロセスの処理がチェックポイントまで終了したら、データ交換処理を行う。一方の並列プロセスの処理が先にチェックポイントに達した場合には、その並列プロセスを実行するCPUは、データ交換相手の並

列プロセスの処理がチェックポイントに達するまで同期待ちを行う。

CPUでのデータ処理には、同期待ち以外にもI/O(Input/Output)待ち等の待ち時間が発生する。待ち状態となったCPUで他のプロセスを実行させれば、システム全体としての処理効率が向上する。そこで、CPUに時分割処理を行わせ、
5 待ち状態のCPUに他のプロセスの処理を実行させることが行われている。ここで他のプロセスは、並列プロセスであってもよいし、単一のCPUで実行される非並列プロセスであってもよい。

なお、並列プロセスのデータ交換の同期待ちの間、CPUが他のプロセスを実行していると、データ交換相手の並列プロセスの処理がチェックポイントに達したときに、他のプロセスの処理が終了していないことがあり得る。すると、データ交換相手の並列プロセスを実行するCPUに、同期待ちの時間が発生してしまう。このような同期待ち時間の発生は、コンピュータシステムの処理効率の低下を招く。
10

そこで、特開平10-74150号公報に記載されたプロセススケジューリング方法では、時分割処理を行うコンピュータシステムの各CPUに、並列プロセスおよび他のプロセスの動作開始・停止を、所定の期間単位（フェーズ）で同時に行わせている。すなわち、ある並列プログラムから生成された複数の並列プロセスは、各CPUにおいて同時に処理が開始され、同時に処理が停止される。これにより、並列プロセス実行時に発生する同期待ちの時間が、時分割処理を行わなかった場合と同じになる。その結果、並列プログラムを構成する並列プロセス間の同期待ちを最小限にし、システム効率の低下を防止することができる。
15
20

ところで、コンピュータで処理するプロセスには、ターンアラウンド時間（プロセスの実行開始から実行終了までの時間）の保証が必要なものがある。ターンアラウンド時間の保証が必要なものとして、たとえば、気象データの解析処理がある。膨大な量の気象データを解析する処理は、気象予報を発表する一定時間前に必ず終了している必要がある。
25

しかし、上記公報記載のプロセススケジューリング方法では、各フェーズの時間が固定であるため、各並列プログラム毎に、ターンアラウンド時間を保証することができない。たとえば、マルチプロセッサ型コンピュータの処理能力の5

0 %を使用しないと、ターンアラウンド時間を保証できないような並列プログラムもあり得る。ところが、上記公報記載のプロセススケジューリング方法では、その並列プログラムの処理に1フェーズ分（たとえば10 %）の時間しか割り当てることができず、ターンアラウンド時間を保証できない。

5

発明の開示

本発明はこのように鑑みてなされたものであり、並列プロセス毎に、任意の割合でCPUの処理時間を配分することができる並列プロセス実行方法、マルチプロセッサ型コンピュータ、並列プロセス実行プログラム、および並列プロセス実行プログラムを記録した記録媒体を提供することを目的とする。

本発明では上記課題を解決するために、図1に示すような並列プロセス実行方法が提供される。本発明の並列プロセス実行方法は、1つの並列プログラムAから生成される複数の並列プロセスA1～A4と他のプロセス（たとえば並列プログラムBの並列プロセスB1～B4）とを、複数のプロセッサ1～4で時分割で処理する場合に適用される。

本発明では、所定の周期の中で、並列プログラムの処理に割り当てべき時間配分率が設定される（ステップS1）。すると、並列プログラムに設定された時間配分率に従って、プロセスの切り替え処理が行われる（ステップS2）。すなわち、並列プログラムAの各並列プロセスA1～A4が複数のプロセッサ1～4中の1つに割り当てられ、複数のプロセッサ1～4において、割り当てられた並列プロセスA1～A4の処理が同時に実行開始される。さらに、並列プロセスA1～A4の実行が開始されてからの経過時間が、所定の周期内での並列プログラムAに設定された時間配分率に応じた時間に達すると、複数のプロセッサ1～4において、割り当てられた並列プロセスの実行が同時に終了する。

これにより、並列プログラムAに対して任意の時間配分率を設定すると、所定の時間中の時間配分率に応じた時間だけ、並列プログラムAから生成された並列プロセスA1～A4の処理に割り当てられる。しかも、並列プログラムAから生成された並列プロセスA1～A4の処理は、複数のプロセッサ1～4において同時に実行開始され、同時に実行終了される。その結果、マルチプロセッサ型コン

コンピュータの処理能力のうち並列プログラムの処理に割り当てるべき割合を任意に設定することができ、ターンアラウンド時間の保証が可能となる。

本発明の上記および他の目的、特徴および利点は本発明の例として好ましい実施の形態を表す添付の図面と関連した以下の説明により明らかになるであろう。

5

図面の簡単な説明

図 1 は、本発明の原理構成図である。

図 2 は、本発明の実施の形態を適用したマルチプロセッサ型コンピュータのハードウェア構成例を示す図である。

10 図 3 は、本実施の形態を実現するための OS の機能を示すブロック図である。

図 4 は、タイムスロット割り当てマップの例を示す図である。

図 5 は、タイムテーブルの例を示す図である。

図 6 は、プロセスの優先度を示す図である。

図 7 は、プロセスの切り替え状況を示すタイムチャートの例である。

15 図 8 は、タイムスロット毎のプロセス切り替え処理を示すフローチャートである。

図 9 は、図 4 に示したタイムスロット割り当てマップに従ったプロセス切り替え状況を示す図である。

図 10 は、時間配分率を均等にした場合のプロセス切り替え例を示す図である。

20 図 11 は、時間配分率を 1 対 9 にした場合のプロセス切り替え例を示す図である。

図 12 は、時間配分率を 2 対 8 にした場合のプロセス切り替え例を示す図である。

25 図 13 は、スループット優先ポリシーにおいて、並列プログラムと非並列プログラムとを時分割で処理する場合のプロセス切り替え例を示す図である。

図 14 は、スループット優先ポリシーにおいて並列プログラム以外に処理すべきプログラムがない場合のプロセス切り替え例を示す図である。

図 15 は、ターンアラウンド優先ポリシーの場合のプロセス切り替え例を示す図である。

図 1 6 は、複数のノードで並列処理を行う場合のシステム構成例を示す図である。

図 1 7 は、複数のノード間でのタイムスロット同期処理のフローチャートである。

- 5 図 1 8 は、ノード間で同期処理が行われた場合の処理プロセスの切り替え状況の例を示す図である。

発明を実施するための最良の形態

以下、本発明の実施の形態を図面を参照して説明する。

- 10 図 1 は、本発明の原理構成図である。本発明に係る並列プロセス実行方法では、並列プログラムを、複数のプロセッサ 1 ～ 4（各プロセッサの識別子は、# 0 ～ # 3）を有するマルチプロセッサ型コンピュータにおいて時分割で処理させる。図 1 の例では、並列プログラム A と並列プログラム B とを時分割で処理する。

- 図 1 に示すように、まず、各並列プログラム A、B に対して、時間配分率が設定される（ステップ S 1）。時間配分率とは、所定の周期（1 サイクル）内で、
15 各並列プログラムから生成されるプロセスがプロセッサを占有する時間の割合である。図 1 の例では、並列プログラム A の時間配分率が 3 0 % に設定されている。また、並列プログラム B の時間配分率が 7 0 % に設定されている。

- 次に、各並列プログラム A、B に対して設定された時間配分率に従って、並列
20 プロセス A 1 ～ A 4 と並列プロセス B 1 ～ B 4 とのプロセス切り替え処理が行われる（ステップ S 2）。プロセス切り替え処理では、並列プロセス A 1 ～ A 4 は、それぞれに対応するプロセッサにより並列に処理される。並列処理では、複数の並列プロセス A 1 ～ A 4 の処理が同時に実行開始される。そして、並列プロセス A 1 ～ A 4 の実行が開始されてからの経過時間が、所定の周期内での並列プログラム A に設定された時間配分率（3 0 %）に応じた時間に達すると、複数のプロ
25 セッサ 1 ～ 4 において並列プロセス A 1 ～ A 4 の実行が同時に終了する。

並列プロセス A 1 ～ A 4 の実行が同時に終了すると、複数のプロセッサ 1 ～ 4 において並列プロセス B 1 ～ B 4 の処理が同時に実行開始される。そして、並列プロセス B 1 ～ B 4 の実行が開始されてからの経過時間が、所定の周期内での並

列プログラムBに設定された時間配分率（70％）に応じた時間に達すると、複数のプロセッサ1～4において並列プロセスB1～B4の実行が同時に終了する。

5 なお、1つの並列プログラムから生成された複数の並列プロセスを実行する過程では、適宜、並列プロセスを実行するプロセッサ同士で、互いにデータ交換が行われる。

10 このように、本発明では時分割処理を行うマルチプロセッサ型コンピュータを用いて、並列プログラムから生成された並列プロセスの処理に割り当てる時間配分率を、任意に設定することができる。そして、1サイクル中の時間配分率に応じた時間を、並列プロセスの処理に割り当てるようにしたため、ターンアラウンドを保証することが可能となる。しかも、1つの並列プログラムから生成された並列プロセスは、同時に実行開始し、同時に実行終了するようにしたため、時分割処理をしたことによるデータ交換のための同期待ちの時間の増加が防止される。

以下に、本発明の実施の形態について具体的に説明する。なお、以下の説明では、プロセッサとしてCPUを用いるものとする。

15 図2は、本発明の実施の形態を適用したマルチプロセッサ型コンピュータのハードウェア構成例を示す図である。コンピュータ10は、複数のCPU11～14によって装置全体が制御されている。各CPU11～14は、バス19を介して互いに接続されている。CPU11～14は、RAM20に格納された共有プログラム等に基づいて生成されたプロセスを実行する。

20 各CPU11～14には、バス19を介して、RAM20、ハードディスク装置（HDD）15、グラフィック処理装置16、入力インタフェース17、および通信インタフェース18が接続されている。

25 RAM20には、CPU11に実行させるOS(Operating System)のプログラムの少なくとも一部や、並列プログラムの少なくとも一部が一時的に格納される。また、RAM20には、タイムスロット割り当てマップ等のデータが格納される。

HDD15は、OSのプログラム、並列プログラム、および非並列プログラムなどが格納される。また、HDD15には、各種プログラムの実行に必要なデータが格納される。

グラフィック処理装置 16 には、モニタ 21 が接続されている。グラフィック処理装置 16 は、CPU 11～14 からの命令に従って、画像をモニタ 21 の画面に表示させる。入力インタフェース 17 には、キーボード 22 とマウス 23 とが接続されている。入力インタフェース 17 は、キーボード 22 やマウス 23 から送られてくる信号を、バス 19 を介して CPU 11～14 に送信する。

通信インタフェース 18 は、ネットワーク 24 に接続されている。ネットワーク 24 は、たとえばインターネットのような広域ネットワークである。通信インタフェース 18 は、ネットワーク 24 を介して、他のコンピュータとの間でデータの送受信を行う。

10 以上のようなハードウェア構成によって、本実施の形態の処理機能を実現することができる。たとえば、図 2 に示したコンピュータの電源が投入されると、HDD 15 に格納された OS のプログラムの一部が、RAM 20 に読み込まれる。そして、各 CPU 11～14 により OS のプログラムが実行される。これにより、各 CPU 11～14 上で OS の動作が開始する。

15 図 3 は、本実施の形態を実現するための OS の機能を示すブロック図である。本実施の形態では、コンピュータ 10 内に OS 30 が立ち上げられる。

OS 30 は、タイムスロット割り当てマップ 31、定義ファイル 32、タイムテーブル 33、GUI 34、プログラム 35、プロセス実行部 36、プロセス切り替え部 37、およびコンテキスト 38 を有している。プロセス実行部 36、プロセス切り替え部 37 及びコンテキスト 38 は、4 つの CPU 11～14 それぞれに設けられている。OS 30 上では、各 CPU 11～14 がそれぞれ識別子 CPU #0、CPU #1、CPU #2、CPU #3 で認識される。

25 タイムスロット割り当てマップ 31 は、プロセスの実行タイミングの協調動作のスケジュールが定義されたデータである。タイムスロット割り当てマップ 31 は、ユーザの操作入力により、任意のデータを登録することが出来る。タイムスロット割り当てマップ 31 には、各 CPU が実行すべきプロセスの割り当てが、タイムスロット毎に登録されている。このタイムスロット割り当てマップ 31 をもとに、各 CPU で現在のタイムスロットにどのプロセスが割り当てられているのかを知ることができる。タイムスロット割り当てマップ 31 の詳細は後述する。

定義ファイル 3 2 は、タイムスロット割り当てマップ 3 1 においてプロセスが未割り当てのタイムスロットをどのように利用するのかのポリシー（判断基準）を示すスケジューリングポリシーの情報を含んでいる。スケジューリングポリシーには、たとえば、スループット優先ポリシーやターンアラウンド優先ポリシーがある。プロセス切り替えに適用するスケジューリングポリシーは、操作入力によって任意に選択することが出来る。

スループット優先ポリシーは、システム全体のスループット（単位時間当りの処理量）向上を優先させるためのポリシーである。スループット優先ポリシーでは、CPU がアイドル状態となる時間をできるだけ減らすように、プロセスが未割り当てのタイムスロット（空きタイムスロット）で何らかのプロセスを実行する。本実施の形態では、実行するプロセスの優先順は、対話型プロセス＞非並列プロセス＞並列プロセスの順とする。

ターンアラウンド優先ポリシーは、ターンアラウンドを保証するためのポリシーである。ターンアラウンド優先ポリシーでは、各プロセスに対して、割り当てられた CPU 配分率以上に CPU を使用させないようにする。すなわち、プロセスが未割り当てのタイムスロットでは、バッチプロセス（並列プロセスまたは非並列プロセス）を実行させない。

タイムテーブル 3 3 は、1 サイクル開始から、各タイムスロットが開始されるまでの時間が設定されたテーブル形式のデータである。タイムテーブル 3 3 の詳細は後述する。

GUI 3 4 は、キーボード 2 2 やマウス 2 3 等の入力装置からの入力信号を解釈し、各種コマンドを生成する。GUI 3 4 は、生成した制御コマンドをプロセス実行部 3 6 に送る。これにより、プロセス実行部 3 6 において対話型プロセスが生成される。また、GUI 3 4 は、プロセス実行部 3 6 から送られる情報を、モニタ 2 1 に表示する。

複数のプログラム 3 5 は、並列プログラムまたは非並列プログラムである。複数の並列プログラムは、CPU で並列処理が可能なプログラムである。非並列プログラムは、1 つの CPU で実行されるプログラムである。複数のプログラム 3 5 は、それぞれの少なくとも一部が、たとえば HDD 1 5 から RAM 2 0 に読み

込まれる。

プロセス実行部 36 は、複数のプログラム 35 や GUI 34 から送られるコマンドに基づいてプロセス（並列プロセスや非並列プロセス）を生成し、生成したプロセスを時分割で実行する。また、プロセス実行部 36 は、実行中の並列プロセスの処理が、データ交換（同期通信）を行うべきチェックポイントに達すると、
5 他の CPU のプロセス実行部との間でデータ交換を行う。

プロセス切り替え部 37 は、プロセス実行部 36 で実行されるプロセスの切り替え制御を行う。具体的には、プロセス切り替え部 37 は、タイムテーブル 33 を参照し、タイムスロットの開始時刻を判断する。プロセス切り替え部 37 は、
10 タイムスロットの開始時刻になると、タイムスロット割り当てマップ 31 を参照し、開始するタイムスロットで実行するプロセスを決定する。

タイムスロット割り当てマップ 31 において、開始するタイムスロットにプロセスが割り当てられていない場合には、プロセス切り替え部 37 は、定義ファイル 32 を参照する。そして、プロセス切り替え部 37 は、定義ファイル 32 に含まれるスケジューリングポリシーに従って、開始するタイムスロットで実行する
15 プロセスを決定する。

プロセス切り替え部 37 は、開始するタイムスロットで実行するプロセスを決定したら、プロセス実行部 36 が行っている処理を中断させ、コンテキストの切り替えを行う。すなわち、プロセス実行部 36 が実行していたプロセスのコンテキストを待避させ、実行するプロセスに対応するコンテキストをプロセス実行部
20 36 に渡す。

コンテキスト 38 は、各プロセスを実行するのに必要な制御情報である。コンテキストには、プロセスが中断したときのプログラムカウンタの内容などが含まれている。

25 図 4 は、タイムスロット割り当てマップの例を示す図である。図 4 の例では、1 サイクルが 10 のタイムスロットに分割されている。各タイムスロットには、タイムスロット番号 #0 ~ #9 が割り振られている。そして、タイムスロット割り当てマップ 31 において、4 つの CPU #0 ~ #3 それぞれに対して、各タイムスロット #0 ~ #9 で実行すべきプロセスが設定されている。

図4の例において、タイムスロット#0～2では、全てのCPU#0～#3に対して、並列プログラムAから生成された並列プロセスが設定されている。タイムスロット#3、#4では、CPU#0、#1に対して、並列プログラムBから生成された並列プロセスが設定されており、CPU#2、#3に対して、並列プログラムCから生成された並列プロセスが設定されている。タイムスロット#5では、CPU#0、#1に対して、並列プログラムDから生成された並列プロセスが設定されており、CPU#2に対して、非並列プログラムEから生成された非並列プロセスが設定されている。なお、タイムスロット#5では、CPU#3にはプロセスが設定されておらず、空きタイムスロットとなっている。タイムスロット#6～#8では、全てのCPU#0～#3に対して、並列プログラムFから生成された並列プロセスが設定されている。タイムスロット#9では、全てのCPU#0～#3に対して、対話型プロセスTが設定されている。なお、対話型プロセスに割り当てられたタイムスロットでは、OS上で生成される複数の対話型プロセスの処理が、優先順位の高い順に実行される。

図5は、タイムテーブルの例を示す図である。タイムテーブル33には、タイムスロット番号毎に、オフセットタイムが設定されている。オフセットタイムとは、1つのサイクルが開始されてから各タイムスロットが開始されるまでの時間である。この例では、1サイクルは1msecであるものとする。

図5の例では、タイムスロット#0のオフセットタイムは「+0msec」である。タイムスロット#1のオフセットタイムは「+100msec」である。タイムスロット#2のオフセットタイムは「+200msec」である。タイムスロット#3のオフセットタイムは「+300msec」である。タイムスロット#4のオフセットタイムは「+400msec」である。タイムスロット#5のオフセットタイムは「+500msec」である。タイムスロット#6のオフセットタイムは「+600msec」である。タイムスロット#7のオフセットタイムは「+700msec」である。タイムスロット#8のオフセットタイムは「+800msec」である。タイムスロット#9のオフセットタイムは「+900msec」である。

各CPUにおいて、1サイクルが開始されてから各タイムスロットのオフセッ

トタイムに等しい時間経過すると、タイムスロット割り当てマップが参照される。そして、切り替え前のタイムスロットにおいて処理するプロセスと、タイムスロット切り替え後において処理するプロセスとが異なる場合には、プロセスの切り替えが行われる。プロセスの切り替えは、優先度を変更することによって行われ

5

図6は、プロセスの優先度を示す図である。優先度は、たとえば60段階程度に分かれている。この例では、最高優先度(H)と、最低優先度(L)とは、バッチプロセス用の優先度である。なお、本実施の形態では、対話型プロセス以外の非並列プロセスおよび並列プロセスは、バッチプロセスとして実行されるものとする。

10

最高優先度(H)より1ランク低い優先度(H-1)から、最低優先度(L)より2ランク高い優先度(L+2)までは、対話型プロセス用の優先度である。最低優先度(L)より1ランク高い優先度(L+1)は、ダミーアイドルプロセスおよび非並列プロセス用の優先度である。ここで、ダミーアイドルプロセスとは、並列プロセスを動作させないために用意されるプロセスである。ダミーアイドルプロセスを実行するCPUは、無意味な処理を繰り返しており、実際には何も処理(計算等)をしない(実効が無い)。

15

各CPUは、タイムスロットの切り替えタイミングにおいて、次のタイムスロットで処理すべきプロセスがあれば、そのプロセスの優先度を、最高優先度(H)に設定する。また、各CPUは、処理対象外のバッチプロセスの優先度を、最低優先度(L)に設定する。

20

図6の例には、CPU#0のタイムスロット#0~#2における各プロセスの優先度を示している。プロセスA1(並列プログラムAから生成された並列プロセス)には、最高優先度(H)が設定されている。対話型プロセスTには、最低優先度(L)より2ランク上の優先度(L+2)が設定されている。並列プロセスB1(並列プログラムBから生成された並列プロセス)、並列プロセスD1(並列プログラムDから生成された並列プロセス)および並列プロセスF1(並列プログラムFから生成された並列プロセス)には、最低優先度(L)が設定されている。

25

時分割処理では、プロセスの切り替えタイミングにおいて、最も優先度の高いプロセスの処理が実行される。したがって、図6の例では、CPU#0においてプロセスA1が実行される。もし、全てのバッチプロセスに対して最低優先度(L)が設定されていれば、対話型プロセスTが実行されることになる。

- 5 図7は、プロセスの切り替え状況を示すタイムチャートの例である。図7には、CPU#0のプロセスの切り替え状況の例を示している。

図7の例では、タイムスロット#0の開始時点において、並列プロセスA1の優先度が最高優先度(H)に設定されている。他の並列プロセスの優先度は、最低優先度(L)である。また、対話型プロセスTの優先度は、最高優先度(H)より1ランク低い優先度(H-1)から、最低優先度(L)より2ランク高い優先度(L+2)の間の優先度が設定されている。その結果、タイムスロット#0では、CPU#0において並列プロセスA1が実行される。

タイムスロット#1の開始時点と、タイムスロット#2との開始時点では、各プロセスの優先度が変化しない。そのため、タイムスロット#1とタイムスロット#2では、引き続き並列プロセスA1が実行される。これにより、1サイクル中の30%の時間が、並列プロセスA1の処理に対して割り当てられる。

タイムスロット#3の開始時点では、並列プロセスA1に対して、最低優先度(L)が設定され、並列プロセスB1に対して、最高優先度(H)が設定される。他の並列プロセスの優先度は変化しない。これにより、タイムスロット#3では、CPU#0において並列プロセスB1が実行される。タイムスロット#4の開始時点では、各プロセスの優先度が変化しない。そのため、タイムスロット#4では、引き続き並列プロセスB1が実行される。これにより、1サイクル中の20%の時間が、並列プロセスB1の処理に対して割り当てられる。

タイムスロット#5の開始時点では、並列プロセスB1に対して、最低優先度(L)が設定され、並列プロセスD1に対して、最高優先度(H)が設定される。他の並列プロセスの優先度は変化しない。これにより、タイムスロット#5では、CPU#0において並列プロセスD1が実行される。これにより、1サイクル中の10%の時間が、並列プロセスD1の処理に対して割り当てられる。

タイムスロット#6の開始時点では、並列プロセスD1に対して、最低優先度

(L) が設定され、並列プロセス F 1 に対して、最高優先度 (H) が設定される。これにより、タイムスロット # 6 では、CPU # 0 において並列プロセス F 1 が実行される。タイムスロット # 7 の開始時点と、タイムスロット # 8 との開始時点では、各プロセスの優先度が変化しない。そのため、タイムスロット # 7 とタイムスロット # 8 では、引き続き並列プロセス F 1 が実行される。これにより、1 サイクル中の 30 % の時間が、並列プロセス F 1 の処理に対して割り当てられる。

タイムスロット # 9 の開始時点では、並列プロセス F 1 に対して、最低優先度 (L) が設定される。他の並列プロセスの優先度は変化しない。これにより、タイムスロット # 9 では、対話型プロセス T の優先度が最も高くなり、CPU # 0 において対話型プロセス T が実行される。これにより、1 サイクル中の 10 % の時間が、全ての対話型プロセス T の処理に対して割り当てられる。

図 8 は、タイムスロット毎のプロセス切り替え処理を示すフローチャートである。この処理は、各タイムスロットの開始時刻になったときに、OS を実行する各 CPU において実行される処理である。以下に、図 8 の処理をステップ番号に沿って説明する。

〔ステップ S 1 1〕 プロセス切り替え部 3 7 (図 3 に示す) は、定義ファイル 3 2 (図 3 に示す) からスケジューリングポリシーの設定を読み出す。

〔ステップ S 1 2〕 プロセス切り替え部 3 7 は、スケジューリングポリシーが、ターンアラウンド優先ポリシーか否かを判断する。ターンアラウンド優先ポリシーであれば、処理がステップ S 1 3 に進められる。ターンアラウンドポリシーでなければ、処理がステップ S 1 4 に進められる。

〔ステップ S 1 3〕 プロセス切り替え部 3 7 は、ダミーアイドルプロセスを生成し、ダミーアイドルプロセスに対して最低優先度より 1 レベル高い優先度 (L + 1) を設定する。

〔ステップ S 1 4〕 プロセス切り替え部 3 7 は、各 CPU に対して同時に割り込みを発生させる。

〔ステップ S 1 5〕 プロセス切り替え部 3 7 は、タイムスロットを 1 つ進める。

〔ステップ S 1 6〕 プロセス切り替え部 3 7 は、タイムスロット割り当てマッ

プ31（図3に示す）を参照し、現タイムスロットが空きか否かを判断する。現タイムスロットが空きであれば、処理がステップS19に進められる。現タイムスロットが空きでなければ、処理がステップS17に進められる。

5 [ステップS17] プロセス切り替え部37は、タイムスロット割り当てマップ31を参照し、現タイムスロットの割り当てが、対話型プロセスか否かを判断する。対話型プロセスが割り当てられていれば、処理がステップS22に進められる。対話型プロセスが割り当てられていなければ、処理がステップS18に進められる。

10 [ステップS18] プロセス切り替え部37は、現タイムスロットに割り当てられたプロセスの優先度を最高値に設定する。その後、処理がステップS22に進められる。

15 [ステップS19] ステップS16において、現タイムスロットが空きと判断された場合には、プロセス切り替え部37は、定義ファイル32を参照し、スループット優先ポリシーか否かを判断する。スループット優先ポリシーの場合には、処理がステップS20に進められる。スループット優先ポリシーではない場合には、処理がステップS22に進められる。

20 [ステップS20] プロセス切り替え部37は、非並列プロセスが存在するか否かを判断する。非並列プロセスが存在する場合には、処理がステップS21に進められる。非並列プロセスが存在しない場合には、処理がステップS22に進められる。

 [ステップS21] プロセス切り替え部37は、非並列プロセスに対して、最低優先度よりも1レベル高い優先度（ $L+1$ ）を設定する。

25 [ステップS22] プロセス切り替え部37は、優先度が最高値の並列プロセスのうち、現タイムスロットに割り当てられたプロセス以外の並列プロセスの優先度を最低値（ L ）にする。

 [ステップS23] プロセス切り替え部37は、タイムテーブル33を参照し、次のプロセッサ割り込みの時刻を設定する。その後、処理がステップS14に進められる。

 このようにして、タイムスロット毎に各プロセスの優先度を変えることができ

る。その結果、タイムスロット割り当てマップおよびスケジューリングポリシーに従ったプロセス切り替えが行われる。

ここで、スケジューリングポリシーがターンアラウンド優先ポリシーの場合、ダミーアイドルプロセス（優先度 $L+1$ ）が生成され（ステップS 1 3）、空き
5 タイムスロットにおいてバッチプロセスの優先度が最低値（ L ）とされる（ステップS 2 2）。そのため、対話型プロセスが存在すればその対話型プロセスが実行され、対話型プロセスが存在しなければ、ダミーアイドルプロセスが実行される。これにより、空きタイムスロットにおいて実効あるバッチプロセスが実行されることはなくなる。

10 また、スケジューリングポリシーがスループット優先ポリシーの場合、空きタイムスロットにおいて、非並列プロセスの優先度が最低値より1ランク上（ $L+1$ ）に設定され（ステップS 2 1）、並列プロセスの優先度が最低値（ L ）とされる（ステップS 2 2）。そのため、対話型プロセス（優先度（ $L+2$ ）～（ $H-1$ ））が存在すればその対話型プロセスが実行され、対話型プロセスが存在し
15 なければ、存在する非並列プロセスが実行される。これにより、対話型プロセス、非並列プロセス、並列プロセスの順番で優先的に実行される。

図9は、図4に示したタイムスロット割り当てマップに従ったプロセス切り替え状況を示す図である。

図9に示すように、タイムスロット#0～タイムスロット#2の期間（1サイクルの30%の時間）、CPU#0において並列プロセスA1が実行され、CPU#1において並列プロセスA2が実行され、CPU#2において並列プロセスA3が実行され、CPU#3において並列プロセスA4が実行される。なお、並列プロセスA1～A4は、共に並列プログラムAから生成されたバッチプロセスである。

25 タイムスロット#3～タイムスロット#4の期間（1サイクルの20%の時間）、CPU#0において並列プロセスB1が実行され、CPU#1において並列プロセスB2が実行され、CPU#2において並列プロセスC1が実行され、CPU#3において並列プロセスC2が実行される。なお、並列プロセスB1、B2は、共に並列プログラムBから生成されたバッチプロセスである。また、並

列プロセスC 1, C 2は、共に並列プログラムCから生成されたバッチプロセスである。

5 タイムスロット# 5の期間（1サイクルの10%の時間）、CPU# 0において並列プロセスD 1が実行され、CPU# 1において並列プロセスD 2が実行され、CPU# 2において非並列プロセスE 1が実行され、CPU# 3においてスケジューリングポリシーに従って決定されたプロセスXが実行される。なお、並列プロセスD 1, D 2は、共に並列プログラムDから生成されたバッチプロセスである。また、非並列プロセスE 1は、非並列プログラムEから生成されたバッチプロセスである。

10 タイムスロット# 6～タイムスロット# 8の期間（1サイクルの30%の時間）、CPU# 0において並列プロセスF 1が実行され、CPU# 1において並列プロセスF 2が実行され、CPU# 2において並列プロセスF 3が実行され、CPU# 3において並列プロセスF 4が実行される。なお、並列プロセスF 1～F 4は、共に並列プログラムFから生成されたバッチプロセスである。

15 タイムスロット# 9の期間（1サイクルの10%の時間）、各CPU# 0～# 3それぞれにおいて、対話型プロセスTが実行される。

以上のように、CPU毎に、各プロセスに割り振る時間配分率を任意に指定できることにより、顧客の要望に応じた時間配分率で、並列プログラムを処理することが出来る。以下に、プロセスに対する時間配分率の設定例を説明する。

20 以下の例では、並列プログラムAは、4つのCPUを常に占有して並列処理を行った場合、処理を終了するのに1日を要するものとする。また、並列プログラムBは、4つのCPUを常に占有して並列処理を行った場合、処理を終了するのに4日を要するものとする。

図10は、時間配分率を均等にした場合のプロセス切り替え例を示す図である。
25 図10の例では、並列プログラムAに対して50%の配分率（タイムスロット# 0～# 4）を割り当て、並列プログラムBに対して50%（タイムスロット# 5～# 9）の配分率を割り当てている。これにより、タイムスロット# 0～# 4の期間、並列プログラムAから生成される並列プロセスA 1～A 4が各CPU# 0～# 3で実行される。また、タイムスロット# 5～# 9の期間、並列プログラム

Bから生成される並列プロセスB 1～B 4が各CPU# 0～# 3で実行される。

図10の場合には、並列プログラムAと並列プログラムBとの処理を同時に実行開始すると、処理の所要時間の短い並列プログラムAの処理が先に終了する。並列プログラムAの処理終了後は、並列プログラムBの処理が100%の配分率
5で行われる。すなわち、並列プログラムAの処理は、処理開始から2日後に終了する。また、並列プログラムBの処理は、処理開始から5日後に終了する。

図11は、時間配分率を1対9にした場合のプロセス切り替え例を示す図である。図11の例では、並列プログラムAに対して10%の配分率（タイムスロット# 0）を割り当て、並列プログラムBに対して90%（タイムスロット# 1～
10# 9）の配分率を割り当てている。これにより、タイムスロット# 0の期間、並列プログラムAから生成される並列プロセスA 1～A 4が各CPU# 0～# 3で実行される。また、タイムスロット# 1～# 9の期間、並列プログラムBから生成される並列プロセスB 1～B 4が各CPU# 0～# 3で実行される。

図11の場合には、並列プログラムAと並列プログラムBとの処理を同時に実行開始すると、処理の所要時間の長い並列プログラムBの処理が先に終了する。
15並列プログラムAの処理終了後は、並列プログラムBの処理が100%の配分率で行われる。すなわち、並列プログラムAの処理は、処理開始から5日後に終了する。また、並列プログラムBの処理は、処理開始から4.4日後に終了する。

このように、並列プログラムに対するCPUの処理時間の配分率を任意に設定
20することで、処理の所要時間の長い並列プログラムBを先に終了させることが可能となる。

図12は、時間配分率を2対8にした場合のプロセス切り替え例を示す図である。図12の例では、並列プログラムAに対して20%の配分率（タイムスロット# 0～# 1）を割り当て、並列プログラムBに対して80%（タイムスロット
25# 2～# 9）の配分率を割り当てている。これにより、タイムスロット# 0～# 1の期間、並列プログラムAから生成される並列プロセスA 1～A 4が各CPU# 0～# 3で実行される。また、タイムスロット# 2～# 9の期間、並列プログラムBから生成される並列プロセスB 1～B 4が各CPU# 0～# 3で実行される。

図12の場合には、並列プログラムAと並列プログラムBとの処理を同時に実行開始すると、並列プログラムAと並列プログラムBとの処理がほぼ同時に終了する。すなわち、並列プログラムAと並列プログラムBとの処理は、共に処理開始から5日後に終了する。

- 5 このように、並列プログラムに対するCPUの処理時間の配分率を任意に設定することで、処理の所要時間の異なる複数の並列プログラムの処理を、同時に終了させることが可能となる。

次に、スケジューリングポリシーの違いに応じたプロセス切り替えの違いについて説明する。

- 10 図13は、スループット優先ポリシーにおいて、並列プログラムと非並列プログラムとを時分割で処理する場合のプロセス切り替え例を示す図である。

ここで、各CPU#0～#3に対して、タイムスロット#0～#4の期間、並列プログラムAの処理を行わせるものとする。CPU#0～#2に対して、タイムスロット#5～#9の期間、並列プログラムBの処理を行わせるものとする。

- 15 そして、CPU#3に対するタイムスロット#5～#9の期間は、空きが設定されているものとする。また、並列プログラムAと並列プログラムB以外に処理すべきプログラムとして非並列プログラムEがあり、対話型プロセスは生成されていないものとする。

- 20 図13の例では、タイムスロット#0～#4の期間、並列プログラムAから生成される並列プロセスA1～A4が各CPU#0～#3で実行される。また、タイムスロット#5～#9の期間、並列プログラムBから生成される並列プロセスB1～B3がCPU#0～#2で実行される。さらに、タイムスロット#5～#9の期間、並列プログラムEから生成される非並列プロセスE1がCPU#3で実行される。

- 25 このように、スループット優先ポリシーを選択し、空きタイムスロットにおいて非並列プロセスを実行すれば、システム全体の効率（スループット）がよくなる。

図14は、スループット優先ポリシーにおいて並列プログラム以外に処理すべきプログラムがない場合のプロセス切り替え例を示す図である。

ここで、各CPU#0～#3に対して、タイムスロット#0～#4の期間、並列プログラムAの処理を行わせるものとする。CPU#0～#2に対して、タイムスロット#5～#9の期間、並列プログラムBの処理を行わせるものとする。そして、CPU#3に対するタイムスロット#5～#9の期間は、空きが設定されているものとする。また、並列プログラムAと並列プログラムB以外に、処理すべきプログラム、あるいは対話型プロセスがないものとする。

図14の例では、タイムスロット#0～#4の期間、並列プログラムAから生成される並列プロセスA1～A4が各CPU#0～#3で実行される。また、タイムスロット#5～#9の期間、並列プログラムBから生成される並列プロセスB1～B3が各CPU#0～#2で実行される。

さらに、タイムスロット#5～#9の期間、並列プログラムAから生成される並列プロセスA4がCPU#3で実行される。しかし、CPU#3でタイムスロット#5～#9の期間、並列プロセスA4の処理実行中に、他の並列プロセスとの間でデータ交換（同期通信）を行うべきチェックポイントに達すると、CPU#3は同期待ちの状態となる。そして、次のサイクルで、他のCPU#0～#2において、並列プロセスA1～A2の処理が進められることで、データ交換が可能となり、同期待ちの状態が解除される。

このように、スループット優先ポリシーにおいて、空きタイムスロットで実行すべき非並列プロセスあるいは対話型プロセスがなければ、並列プロセスの処理が進められる。これにより、システム全体の効率（スループット）はよくなる。ただし、並列プロセスの同期待ちの時間は、その並列プロセスの処理中の時間として換算される。すなわち、サーバコンピュータの処理機能を顧客に使用させ、CPUの動作時間に応じて料金を徴収していた場合、同期待ちのCPUループの時間も課金されてしまう。すると、同じ並列プログラムを実行した場合であっても、空きのタイムスロットがあるか否かにより料金が異なってしまう。

そこで、顧客が、並列プログラムの内容が同じであれば同じ料金が徴収されることを望む場合には、ターンアラウンド優先ポリシーが選択される。また、顧客によりターンアラウンドの保証が望まれる場合にも、ターンアラウンド優先ポリシーが選択される。

図15は、ターンアラウンド優先ポリシーの場合のプロセス切り替え例を示す図である。

ここで、各CPU#0～#3に対して、タイムスロット#0～#4の期間、並列プログラムAの処理を行わせるものとする。CPU#0～#2に対して、タイムスロット#5～#9の期間、並列プログラムBの処理を行わせるものとする。そして、CPU#3に対するタイムスロット#5～#9の期間は、空きが設定されているものとする。

図15の例では、タイムスロット#0～#4の期間、並列プログラムAから生成される並列プロセスA1～A4が各CPU#0～#3で実行される。また、タイムスロット#5～#9の期間、並列プログラムBから生成される並列プロセスB1～B3が各CPU#0～#2で実行される。さらに、タイムスロット#5～#9の期間、ダミーアイドルプロセスGがCPU#3で実行される。

ダミーアイドルプロセスGが実行されている時間は、並列プロセスの処理時間として換算されない。従って、並列プログラムが各CPUを占有する時間の総計は、空きのタイムスロットの有無に依存しない。その結果、並列プログラムの内容が同じであれば同じ料金が徴収される。

次に、複数のコンピュータ（以下、ノードという）で並列処理を行う場合の例について説明する。

図16は、複数のノードで並列処理を行う場合のシステム構成例を示す図である。図16の例では、協働制御装置100に対して、割り込み通知装置201、301を介して、複数のノード200、300が接続されている。ノード200は、複数のCPU211、212を有している。同様に、ノード300は、複数のCPU311、312を有している。

協働制御装置100は、各ノード間での協働動作を制御する。たとえば、協働制御装置100は、各ノード間で交換されるデータを中継する。また、協働制御装置100は、システムの起動時に、各ノードに対して同期用の割り込み信号を通知する。

割り込み通知装置201、301は、協働制御装置100からの割り込み信号を受け取り、ノード内の所定のCPUに対して、割り込み信号を通知する。たと

えば、ノード２００に接続された割り込み通知装置２０１は、ＣＰＵ２１１に対して割り込み信号を通知する。ノード３００に接続された割り込み通知装置３０１は、ＣＰＵ３１１に対して割り込み信号を通知する。

- 各ノード２００、３００では、任意のＣＰＵ２１１、３１１が割り込み信号の通知を受け取ると、タイムスロットの初期化やノード内の各ＣＰＵ割り込みの設定が行われる。

- 図１７は、複数のノード間でのタイムスロット同期処理のフローチャートである。以下に、図１７の処理をステップ番号に沿って説明する。なお、図１７の例では、ノード２００に接続された割り込み通知装置２０１を用いて、割り込みタイマの設定を行うものとする。

〔ステップＳ３１〕ノード２００は、割り込み通知装置２０１に対して、並列処理の開始時刻を通知する。具体的には、ユーザからの操作入力に応答して、ノード２００で実行されているＯＳから割り込み通知装置２０１に対して開始時刻が通知される。

- 〔ステップＳ３２〕割り込み通知装置２０１は、協働制御装置１００に対して開始時刻を通知する。これにより、協働制御装置１００において、開始時刻を割り込み発生時刻としたタイマが設定される。

- 〔ステップＳ３３〕協働制御装置１００は、タイマ設定に基づいて、開始時刻に達したか否かを判断する。開始時刻に達していれば、処理がステップＳ３４に進められる。開始時刻に達していなければステップＳ３３の処理が繰り返される。

〔ステップＳ３４〕協働制御装置１００は、タイマに設定された開始時刻に達すると、協働制御装置１００に接続された全ての割り込み通知装置２０１、３０１に対してタイマ割り込みを通知する。

- 〔ステップＳ３５〕タイマ割り込みを受け取った各割り込み通知装置２０１、３０１は、それぞれに対応するノードの所定のＣＰＵに対して、タイマ割り込みを通知する。

〔ステップＳ３６〕タイマ割り込みの通知を受けた各ＣＰＵは、あらかじめ用意されている割り込みハンドラの処理ルーチン（プログラム）を起動する。すると、割り込みハンドラの処理に従って、タイムスロットの初期化（現行のタイム

スロット番号を# 0にする処理等)と、CPU割り込み設定(所定の時刻にノード内の全てのCPUに対して同時に発生させる割り込み信号の設定)が行われる。CPU割り込みの設定は、全てのノードの全てのCPUに対して、同時刻に割り込みが発生するように設定される。

- 5 このように、全てのCPUに対する同時刻の割り込み設定が行われることにより、各ノードの各CPUにおいて、同時に割り込みが発生する。そして、タイムスロットの切り替え処理のサイクルが開始される。

図18は、ノード間で同期処理が行われた場合の処理プロセスの切り替え状況の例を示す図である。図18の例では、ノード#0とノード#1とを同時に起動した場合の、各CPUにおける処理を示している。ノード#0とノード#1とが
10 同時に起動されると、それぞれのCPUにおいて、起動処理が実行される。起動処理としては、たとえば、セルフテストの実施や、所定のブートプログラムのロード等の処理がある。起動処理の終了時刻は、CPUによって異なる。起動処理が終了したCPUは、CPUの割り込み発生を待つ。

- 15 全てのCPUにおいて起動処理が終了した後、全てのCPUに対して同時に、割り込みを発生させる。各CPUは、割り込みの発生後、タイムスロットの切り替え処理を開始し、タイムスロット毎に、タイムスロット割り当てマップに設定されたプロセスを実行する。

このように、複数のノードにおいて、タイムスロットの切り替え処理のタイミ
20 ングを同期させることにより、並列プログラムが生成される複数の並列プロセスを、複数のノードで分散して並列処理することが可能となる。すなわち、タイムスロットの切り替え処理のタイミングを同期させない場合、並列プログラムAの各並列プロセスが別々のノードで異なる時間帯に実行される場合が発生し得る。この場合、異なるノードで処理されている複数の並列プロセス間において、デー
25 タ交換のための同期待ち時間が過大になる虞がある。

ところが、タイムスロットの切り替え処理のタイミングを同期させることで、複数のノードで分散して処理される各並列プロセスが同時に実行開始され、同時に実行が終了される。その結果、データ交換時の同期待ち時間を最小限に抑えることが可能となる。

以上説明したように、本発明の実施の形態によれば、並列プログラムを処理する際のCPUの時間配分率を任意に設定できることで、ターンアラウンド時間を保証することが可能となる。ターンアラウンド時間を保証することで、定期的に同じプログラムを実行するような場合に、他のプログラムの動作の有無に関係なく、いつも同じ時間で処理を終了させることができる。処理の終了時間の予測ができれば、処理結果を利用した作業を、予定通りに進めることができる。たとえば、気象予報のデータ処理において、システムの処理負荷に関係なく、常に予報の発表時刻よりも一定時間前に、気象データの解析を終了させることができる。

また、並列プログラムを処理する際のCPUの時間配分率を任意に設定できることは、バッチプロセスごとにランク分けが可能となることを意味する。すなわち、重要なバッチプロセスには、多くのCPU配分率を与える。これにより、たとえば、重要でないバッチプロセスよりも早く重要なバッチプロセスの処理を終了させることができる。また、他のバッチプロセスよりも処理の所要時間が長いバッチプロセスの処理を、他のバッチプロセスの処理と同時に終了させることもできる。

また、対話型プロセス用のタイムスロットを設定出来るようにしたことにより、一定時間間隔で、確実に対話型プロセスを実行させることができる。すなわち、全てのタイムスロットに対してバッチプロセス（たとえば並列プロセス）を割り当てると、バッチプロセスが終了するまで、各タイムスロットにおいてバッチプロセスが最優先で実行される。すると、バッチプロセスが終了するまで、対話型プロセスが実行出来なくなってしまう。そこで、本実施の形態では、対話型プロセス用のタイムスロットにおいては、全てのバッチプロセスの優先度を、最低の値で設定するようにした。これにより、対話型プロセス全体へのCPUの時間配分率が保証できる。

また、本実施の形態では、スケジューリングポリシーによって、スループット優先ポリシーとターンアラウンド優先ポリシーとのどちらかを、任意に選択できるようにした。これにより、顧客のニーズに応じて、スループットを優先したり、ターンアラウンドを優先したりすることができる。すなわち、本実施の形態では、プロセスが割当てられていないタイムスロットで、並列プロセスを動作させるか

否かを、システム単位で指定することが可能となった。

たとえば、課金を気にせずに、頻繁に同期をとらない並列プロセスが主に実行されるようなシステムではスループット優先ポリシーを選択し、何も割当てられていないタイムスロットでいずれかのプロセスを動作させる。これにより、システム全体でのスループットが向上する。しかも、空きタイムスロットで実行するプロセスを選択する際の優先順位を、対話型プロセス、非並列プロセス、並列プロセスの順にしたため、空きタイムスロットで実行される処理において同期待ちが発生する可能性がさらに低くなる。

一方、頻繁に同期をとる並列プロセスが主に実行されるようなシステムでは、何も割当てられていない空きタイムスロットで並列プロセスを実行しても、CPUの処理能力が無駄に消費されるだけである。この場合、空きタイムスロットで並列プロセスを実行しても、システム全体のスループットが向上しないばかりか、課金にもブレが生じてしまう。このような場合は、ターンアラウンド優先ポリシーに設定することで、何も割り当てられていないタイムスロットでは並列プロセスを実行しないようにすることで、CPUの処理能力が無駄に消費されるのを防止できる。

さらに、複数のCPUが複数のノードに分散している場合、各ノードのCPUで実行されるタイムスロットのサイクルを同期させるようにした。これにより、1つの並列プログラムから生成される複数の並列プロセスを、複数のノードで並列に処理しても、1つのノード内で並列処理したときと同程度の同期待ちしか発生しない。すなわち、複数のノードで並列処理することによる処理効率の低下を防止することが出来る。

なお、上記の例では、マルチCPUにおける並列処理において、プロセスの実行タイミングをスケジュールに合わせて協調動作させるように説明したが、1つのノード内であれば、複数のスレッド間での協調動作をスケジューリングすることもできる。すなわち、タイムスロットの割り当てマップにおいて、スレッド単位で各CPUの処理対象を割り当てることも可能である。

なお、上記の処理内容は、コンピュータで読み取り可能な記録媒体に記録されたプログラムに記述されており、このプログラムをコンピュータで実行すること

により、上記処理がコンピュータで実現される。コンピュータで読み取り可能な記録媒体としては、磁気記録装置や半導体メモリ等がある。市場へ流通させる場合には、CD-ROM(Compact Disk Read Only Memory)やフロッピーディスク等の可搬型記録媒体にプログラムを格納して流通させることができる。また、

5 ネットワークを介して接続されたコンピュータの記憶装置にプログラムを格納しておき、ネットワークを介して接続された他のコンピュータにプログラムを転送することもできる。コンピュータで実行する際には、コンピュータ内のハードディスク装置等にプログラムを格納しておき、メインメモリにロードして実行することができる。

- 10 以上説明したように本発明では、並列プログラムに対して任意の時間配分率が設定されると、並列プロセスの処理を複数のプロセッサにおいて同時に実行開始し、所定の時間中の時間配分率に応じた時間経過後に、並列プロセスの処理を同時に実行終了するようにした。これにより、並列プログラムを処理する際のプロセッサの処理時間の時間配分率を任意に設定し、ターンアラウンド時間を保証する
- 15 ことが可能となる。

上記については単に本発明の原理を示すものである。さらに、多数の変形、変更が当業者にとって可能であり、本発明は上記に示し、説明した正確な構成および応用例に限定されるものではなく、対応するすべての変形例および均等物は、添付の請求項およびその均等物による本発明の範囲とみなされる。

請 求 の 範 囲

1. 1つの並列プログラムから生成される複数の並列プロセスと他のプロセスとを、複数のプロセッサで時分割で処理するための並列プロセス実行方法において、

所定の周期の中で前記並列プログラムの処理に割り当てるべき時間配分率を設定し、

前記並列プログラムの前記各並列プロセスを前記複数のプロセッサ中の1つに割り当て、前記複数のプロセッサにおいて、割り当てられた並列プロセスの処理を同時に実行開始し、

前記並列プロセスの実行が開始されてからの経過時間が、前記所定の周期内での前記並列プログラムに設定された前記時間配分率に応じた時間に達すると、前記複数のプロセッサにおいて、割り当てられた並列プロセスの実行を同時に終了する、

15 ことを特徴とする並列プロセス実行方法。

2. 前記所定の周期を複数のタイムスロットに分割し、各プロセッサにおいて各タイムスロットで実行すべきプロセスを設定することにより、前記並列プログラムの処理に割り当てるべき時間配分率を設定することを特徴とする請求項1記載の並列プロセス実行方法。

20 3. 前記各タイムスロットで実行すべきプロセスには、対話型プロセスが含まれることを特徴とする請求項2記載の並列プロセス実行方法。

4. プロセスが設定されていない空きタイムスロットにおける処理対象の判断基準があらかじめ定義されており、前記各プロセッサは、前記空きタイムスロットにおいては、前記判断基準に従って、処理対象となるプロセスを選択することを特徴とする請求項2記載の並列プロセス実行方法。

25 5. 前記判断基準には、前記空きタイムスロットにおいて、バッチプロセスを実行可能なスループット優先ポリシーと、前記空きタイムスロットにおいて、バッチプロセスを実行しないターンアラウンド優先ポリシーとがあらかじめ用意されており、前記スループット優先ポリシーと前記ターンアラウンド優先ポリシー

とから任意に選択できることを特徴とする請求項4記載の並列プロセス実行方法。

6. 前記スループット優先ポリシーでは、プロセス選択の優先順位の高い方から、対話型プロセス、単一のプロセッサで実行される非並列プロセス、並列プロセスの順番であることを特徴とする請求項5記載の並列プロセス実行方法。

- 5 7. 前記複数のプロセッサが複数のノードに分散して配置されている場合には、各ノードに対して同時に割り込みを発生させ、当該割り込みを前記各ノードが有する各プロセッサに同時に通知し、割り込み発生のタイミングに合わせて前記各プロセッサによる前記所定の周期の計時を開始することを特徴とする請求項1記載の並列プロセス実行方法。

- 10 8. 1つの並列プログラムから生成される複数の並列プロセスと他のプロセスとを時分割で処理するための複数のプロセッサを有するマルチプロセッサ型コンピュータにおいて、

所定の周期の中で前記並列プログラムの処理に割り当てるべき時間配分率を設定する時間配分率設定手段と、

- 15 前記並列プログラムの前記各並列プロセスを前記複数のプロセッサ中の1つに割り当て、前記複数のプロセッサにおいて、割り当てられた並列プロセスの処理を同時に実行開始させ、前記並列プロセスの実行が開始されてからの経過時間が、前記所定の周期内での前記並列プログラムに設定された前記時間配分率に応じた時間に達すると、前記複数のプロセッサにおいて、それぞれに割り当てられた並列プロセスの実行を同時に終了させるプロセス実行手段と、
- 20

を有することを特徴とするマルチプロセッサ型コンピュータ。

9. 前記時間配分率設定手段では、前記所定の周期を複数のタイムスロットに分割し、各プロセッサにおいて各タイムスロットで実行すべきプロセスを設定することを特徴とする請求項8記載のマルチプロセッサ型コンピュータ。

- 25 10. プロセスが設定されていない空きタイムスロットにおける各プロセッサの処理対象の判断基準があらかじめ定義されており、

前記プロセス実行手段は、前記各プロセッサに対して、前記空きタイムスロットにおいては、前記判断基準に従って処理対象となるプロセスを選択させることを特徴とする請求項9記載のマルチプロセッサ型コンピュータ。

1 1. 前記判断基準には、前記空きタイムスロットにおいて、バッチプロセスを実行可能なスループット優先ポリシーと、前記タイムスロットにおいて、バッチプロセスを実行しないターンアラウンド優先ポリシーとがあらかじめ用意されており、前記スループット優先ポリシーと前記ターンアラウンド優先ポリシーとから任意に選択できることを特徴とする請求項10記載のマルチプロセッサ型コンピュータ。

1 2. 1つの並列プログラムから生成される複数の並列プロセスと他のプロセスとを時分割で処理するための複数のプロセッサが、複数のノードに分散して配置されたマルチプロセッサ型コンピュータにおいて、

10 前記複数のノードそれぞれに対して、割り込み通知を同時に発生させる協働制御装置と、

所定の周期の中で前記並列プログラムの処理に割り当てるべき時間配分率を設定する時間配分率設定手段と、

前記並列プログラムの前記各並列プロセスを前記複数のプロセッサ中の1つに
15 割り当て、前記複数のプロセッサにおいて、割り当てられた並列プロセスの処理を同時に実行開始させ、前記並列プロセスの実行が開始されてからの経過時間が、前記所定の周期内での前記並列プログラムに設定された前記時間配分率に応じた時間に達すると、前記複数のプロセッサにおいて、それぞれに割り当てられた並列プロセスの実行を同時に終了させる、前記複数のノードそれぞれに設けられた
20 複数のプロセス実行手段と、

を有することを特徴とするマルチプロセッサ型コンピュータ。

1 3. 1つの並列プログラムから生成される複数の並列プロセスと他のプロセスとを、複数のプロセッサで時分割で処理するための並列プロセス実行プログラムにおいて、

25 コンピュータに、

所定の周期の中で前記並列プログラムの処理に割り当てるべき時間配分率を設定し、

前記並列プログラムの前記各並列プロセスを前記複数のプロセッサ中の1つに割り当て、前記複数のプロセッサにおいて、割り当てられた並列プロセスの処理

を同時に実行開始し、

前記並列プロセスの実行が開始されてからの経過時間が、前記所定の周期内での前記並列プログラムに設定された前記時間配分率に応じた時間に達すると、前記複数のプロセッサにおいて、それぞれに割り当てられた並列プロセスの実行を

5 同時に終了する、

処理を実行させることを特徴とする並列プロセス実行プログラム。

1 4. 1つの並列プログラムから生成される複数の並列プロセスと他のプロセスとを、複数のプロセッサで時分割で処理するためのプログラムを記録したコンピュータ読み取り可能な記録媒体において、

10 前記コンピュータに、

所定の周期の中で、前記並列プログラムの処理に割り当てべき時間配分率を設定し、

前記並列プログラムの前記各並列プロセスを前記複数のプロセッサ中の1つに割り当て、前記複数のプロセッサにおいて、割り当てられた並列プロセスの処理

15 を同時に実行開始し、

前記並列プロセスの実行が開始されてからの経過時間が、前記所定の周期内での前記並列プログラムに設定された前記時間配分率に応じた時間に達すると、前記複数のプロセッサにおいて、それぞれに割り当てられた並列プロセスの実行を同時に終了する、

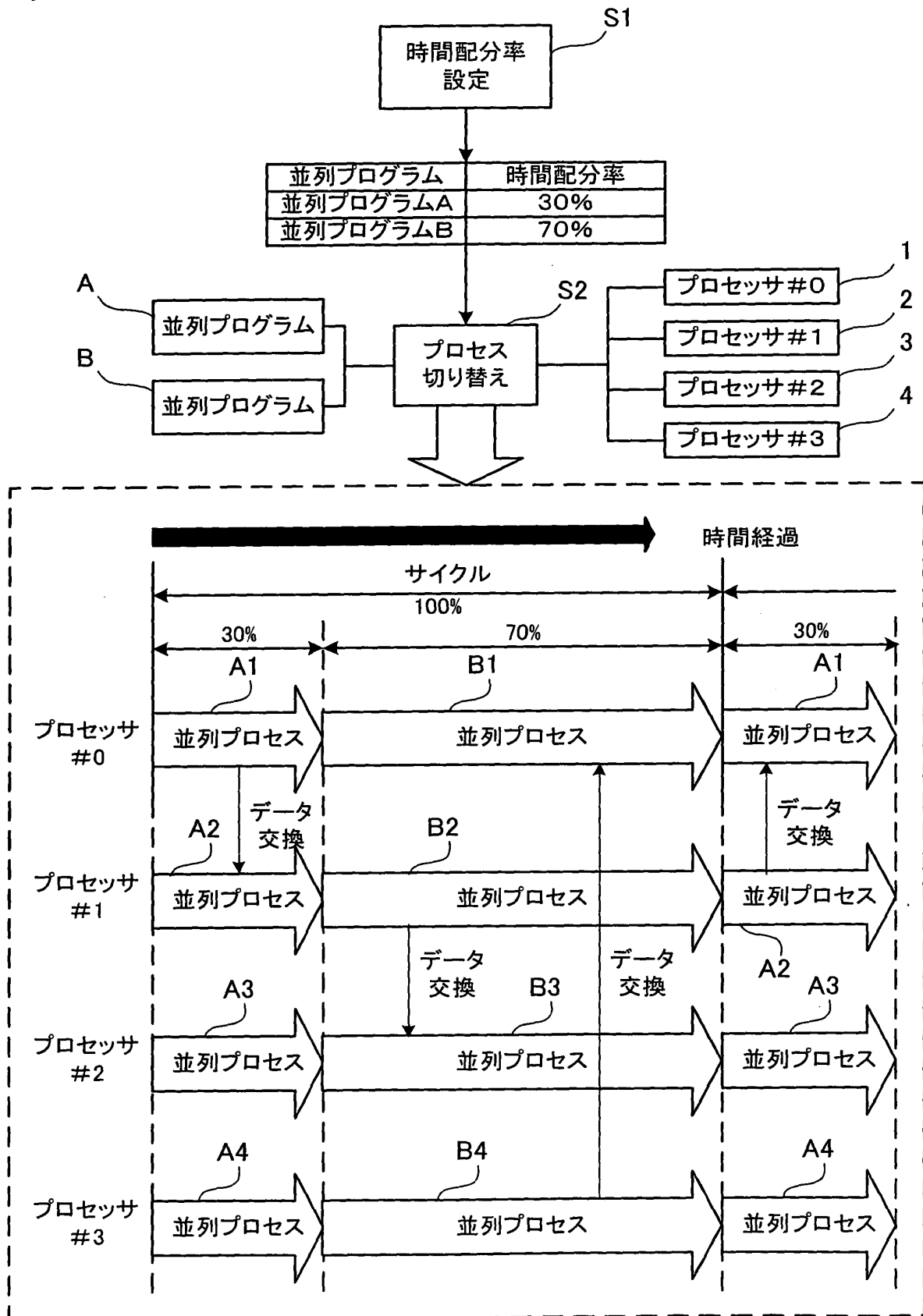
20 処理を実行させることを特徴とする記録媒体。

要 約 書

並列プロセス毎に、任意の割合でCPUの処理時間を配分することができるようにする。並列プログラムの処理に割り当てべき時間配分率が設定される（ステップS1）。すると、並列プログラムに設定された時間配分率に従って、プロセスの切り替え処理が行われる（ステップS2）。すなわち、並列プログラム（A）の各並列プロセス（A1）～（A4）が複数のプロセッサ（1）～（4）中の1つに割り当てられ、複数のプロセッサ（1）～（4）において、割り当てられた並列プロセス（A1）～（A4）の処理が同時に実行開始される。さらに、並列プロセス（A1）～（A4）の実行が開始されてからの経過時間が、所定の周期内での並列プログラム（A）に設定された時間配分率に応じた時間に達すると、複数のプロセッサ（1）～（4）において、割り当てられた並列プロセスの実行が同時に終了する。

図 1

1/18



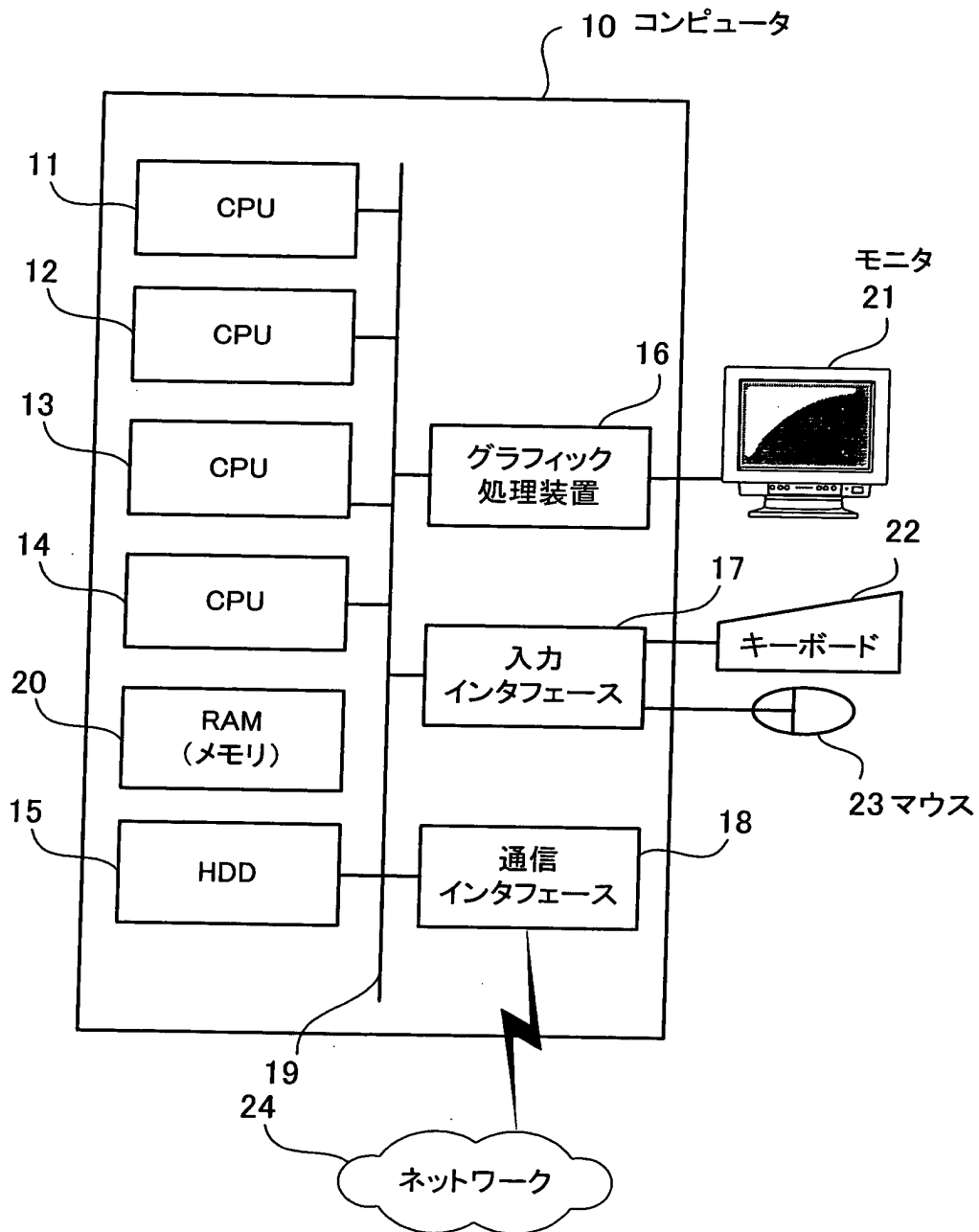


図2

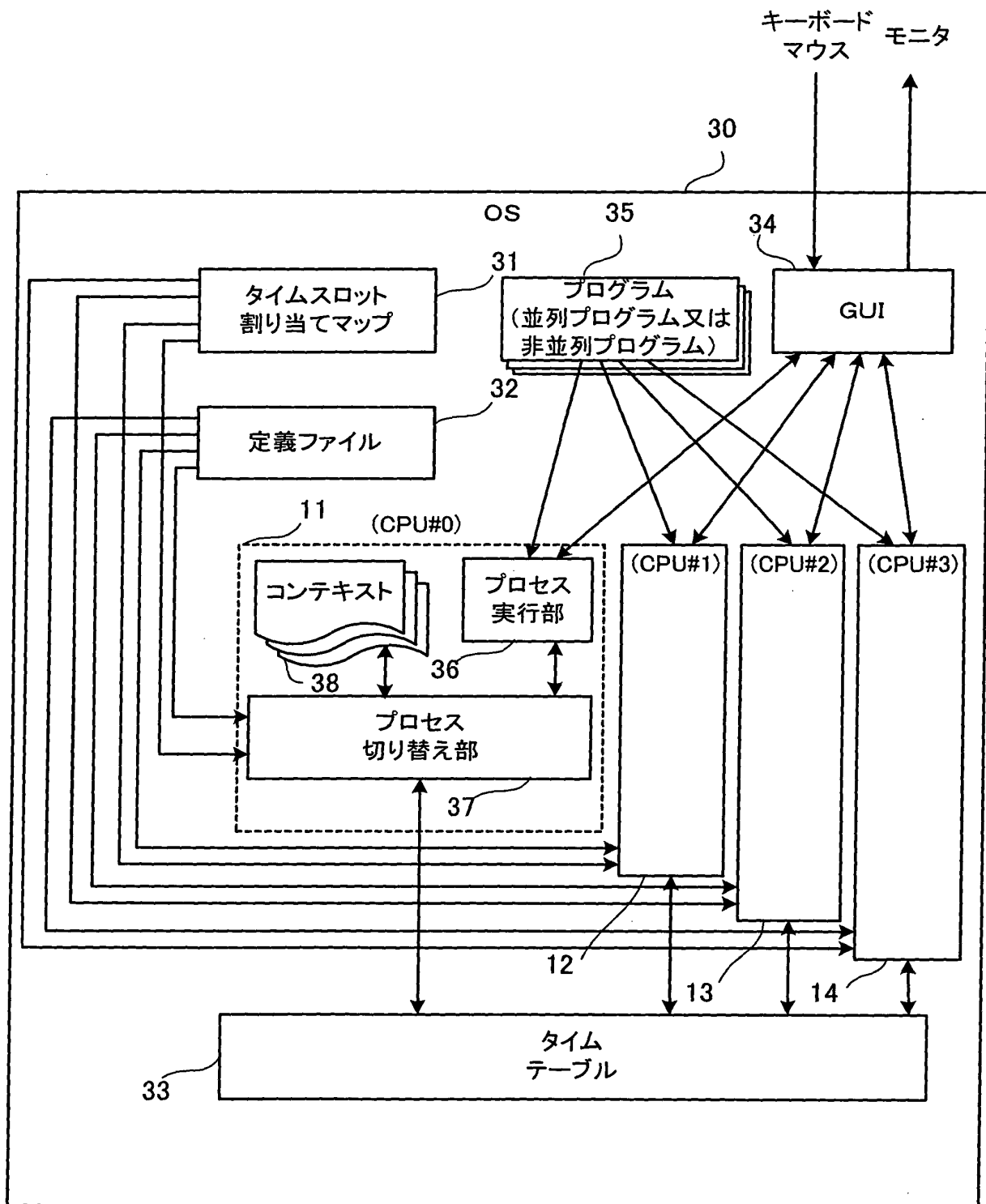


図3

31 タイムスロット割り当てマップ

タイムスロット番号										
	#0	#1	#2	#3	#4	#5	#6	#7	#8	#9
CPU#0	A	A	A	B	B	D	F	F	F	T
CPU#1	A	A	A	B	B	D	F	F	F	T
CPU#2	A	A	A	C	C	E	F	F	F	T
CPU#3	A	A	A	C	C	空	F	F	F	T

図4

33 タイムテーブル

タイムスロット番号	0	1	2	3	4	5	6	7	8	9
オフセットタイム 格納領域	+0msec	+100msec	+200msec	+300msec	+400msec	+500msec	+600msec	+700msec	+800msec	+900msec

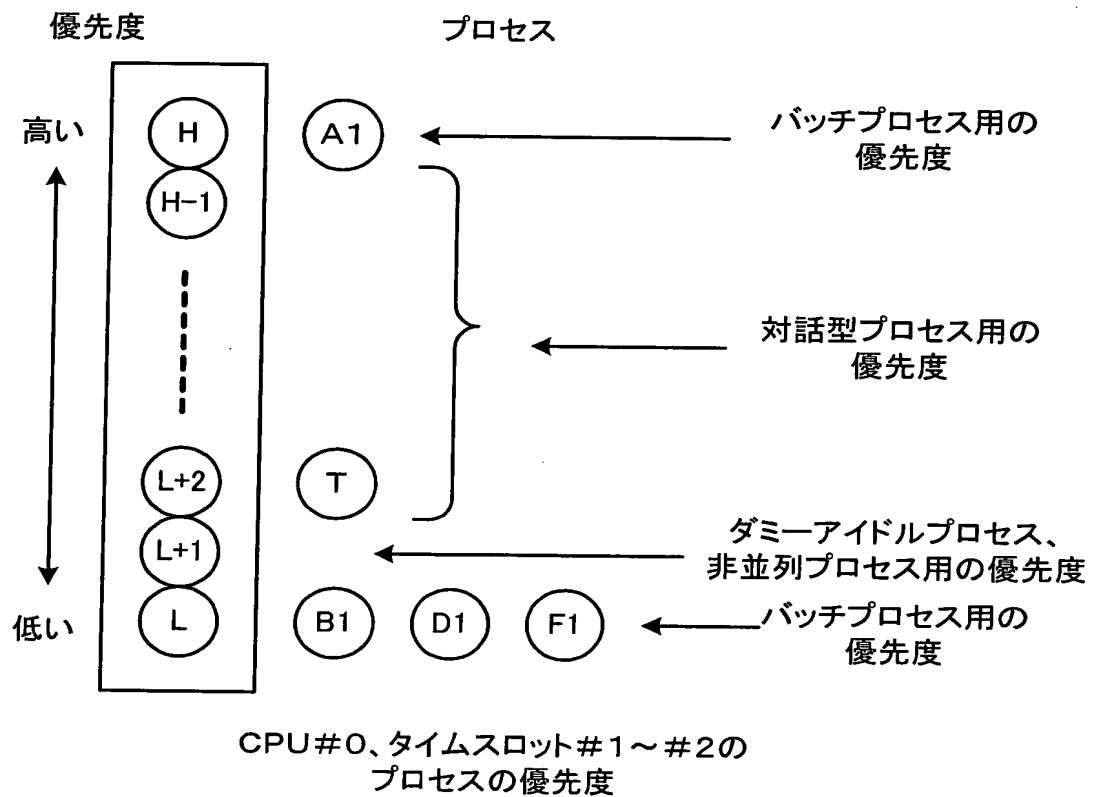
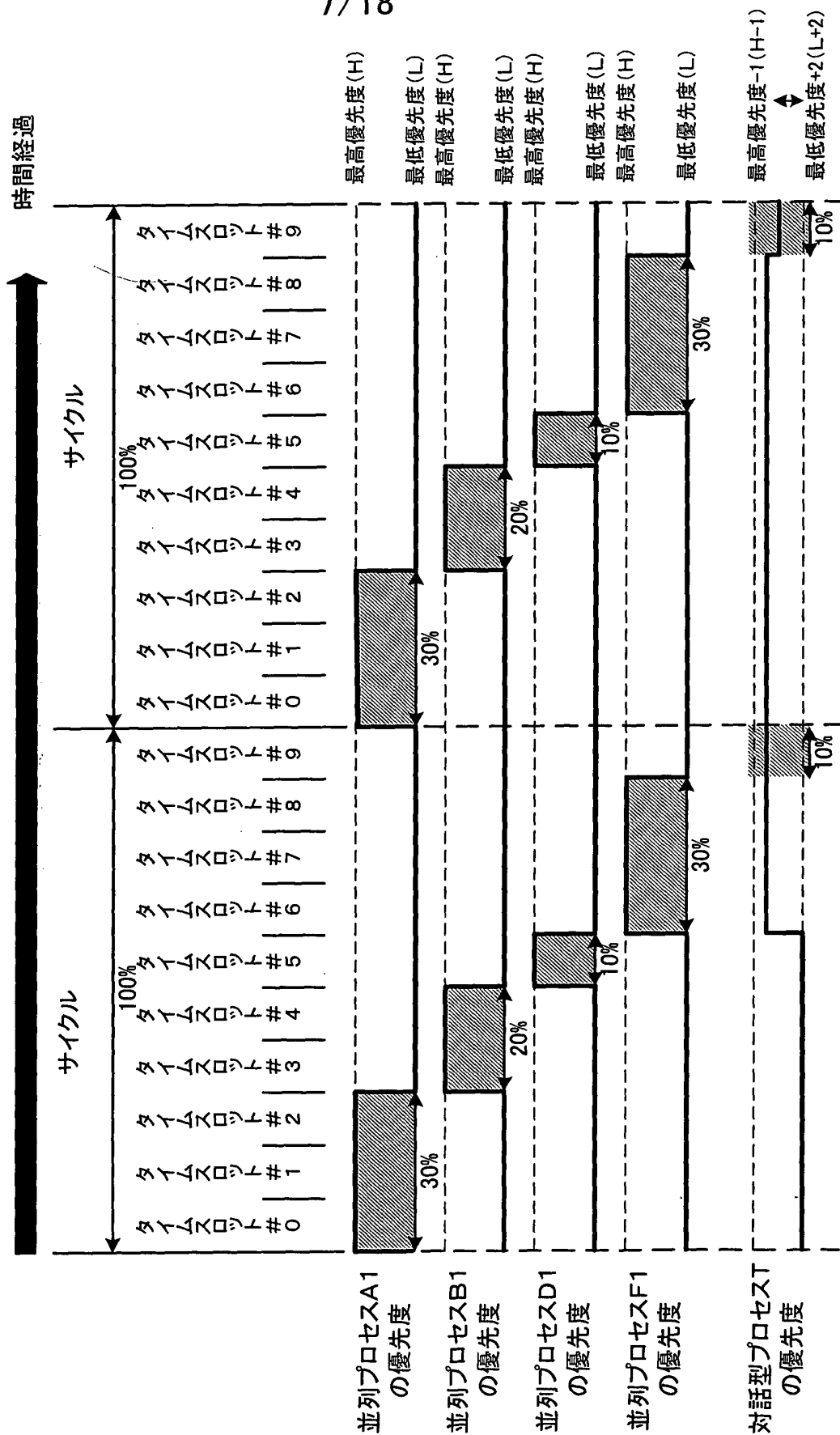
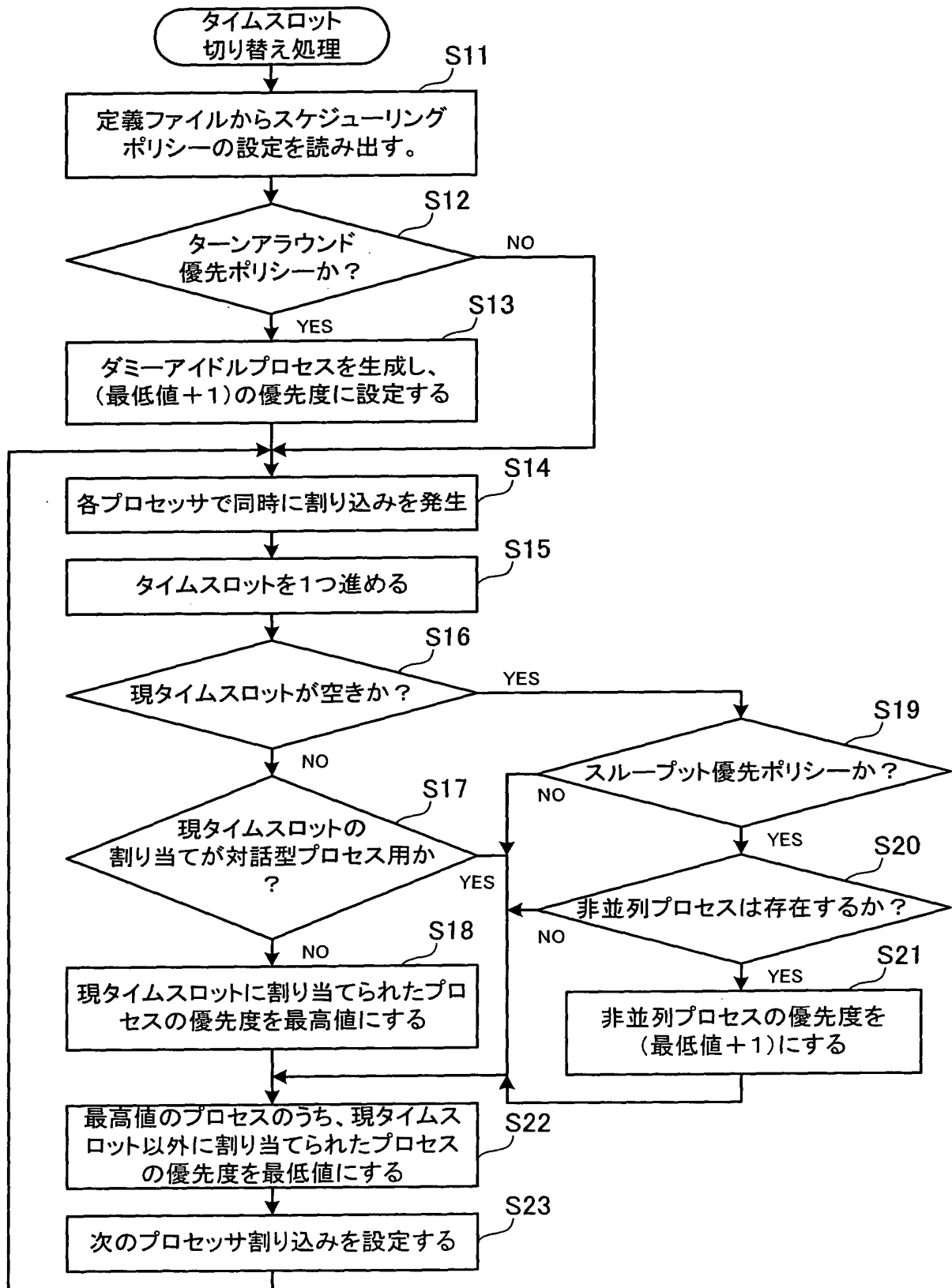


図6

図7

CPU #0のプロセス切り替え状況





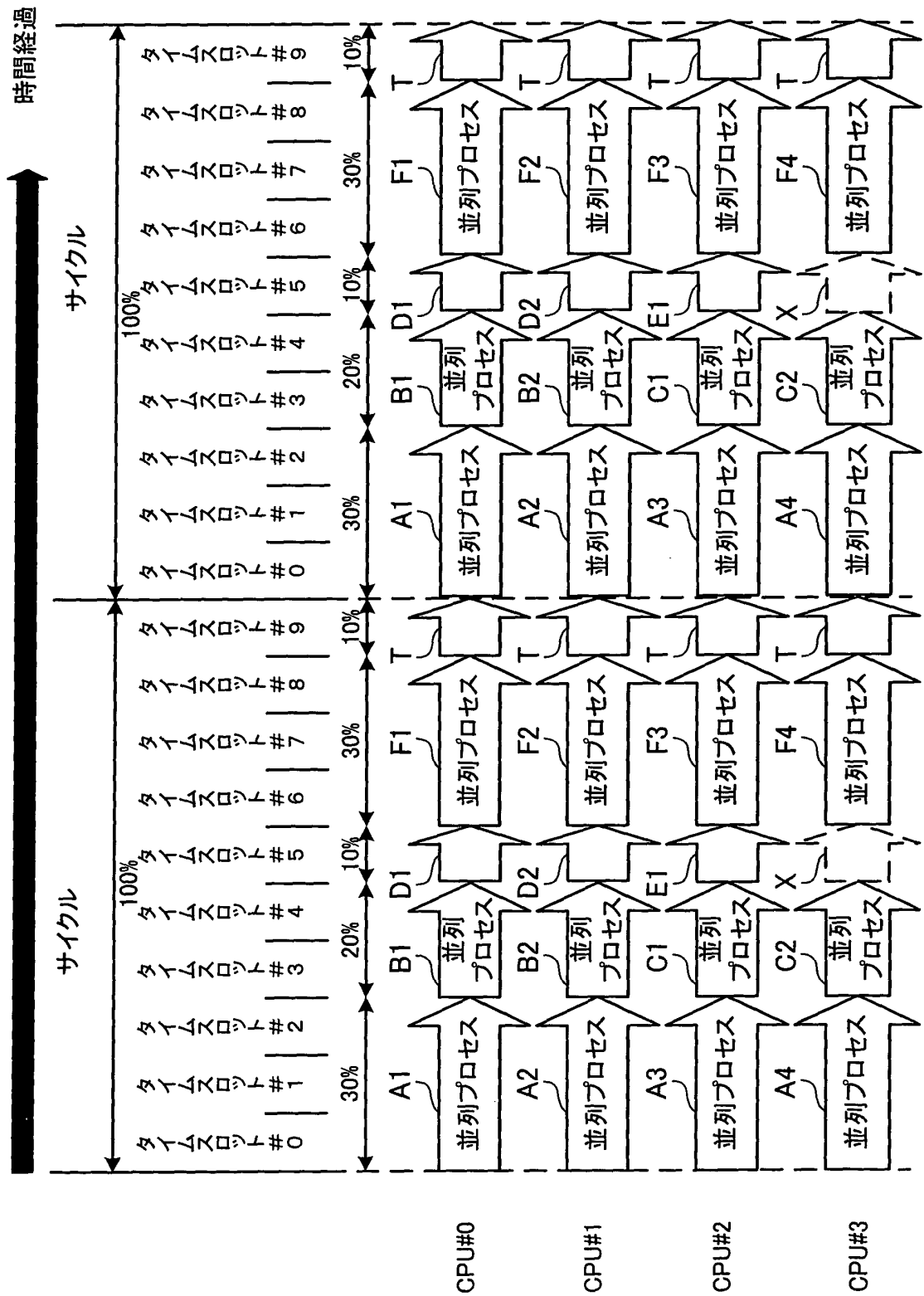


図10

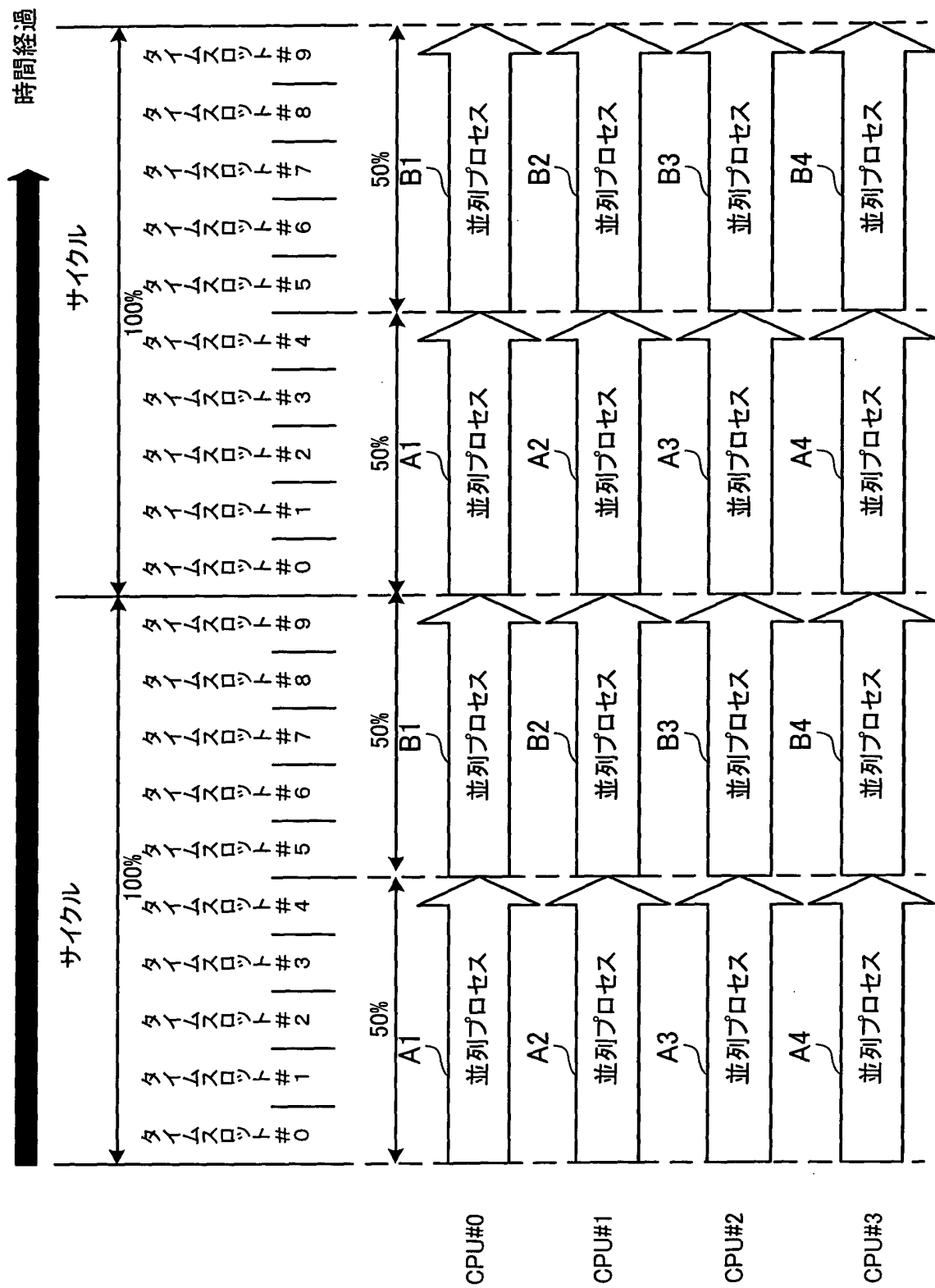


図 11

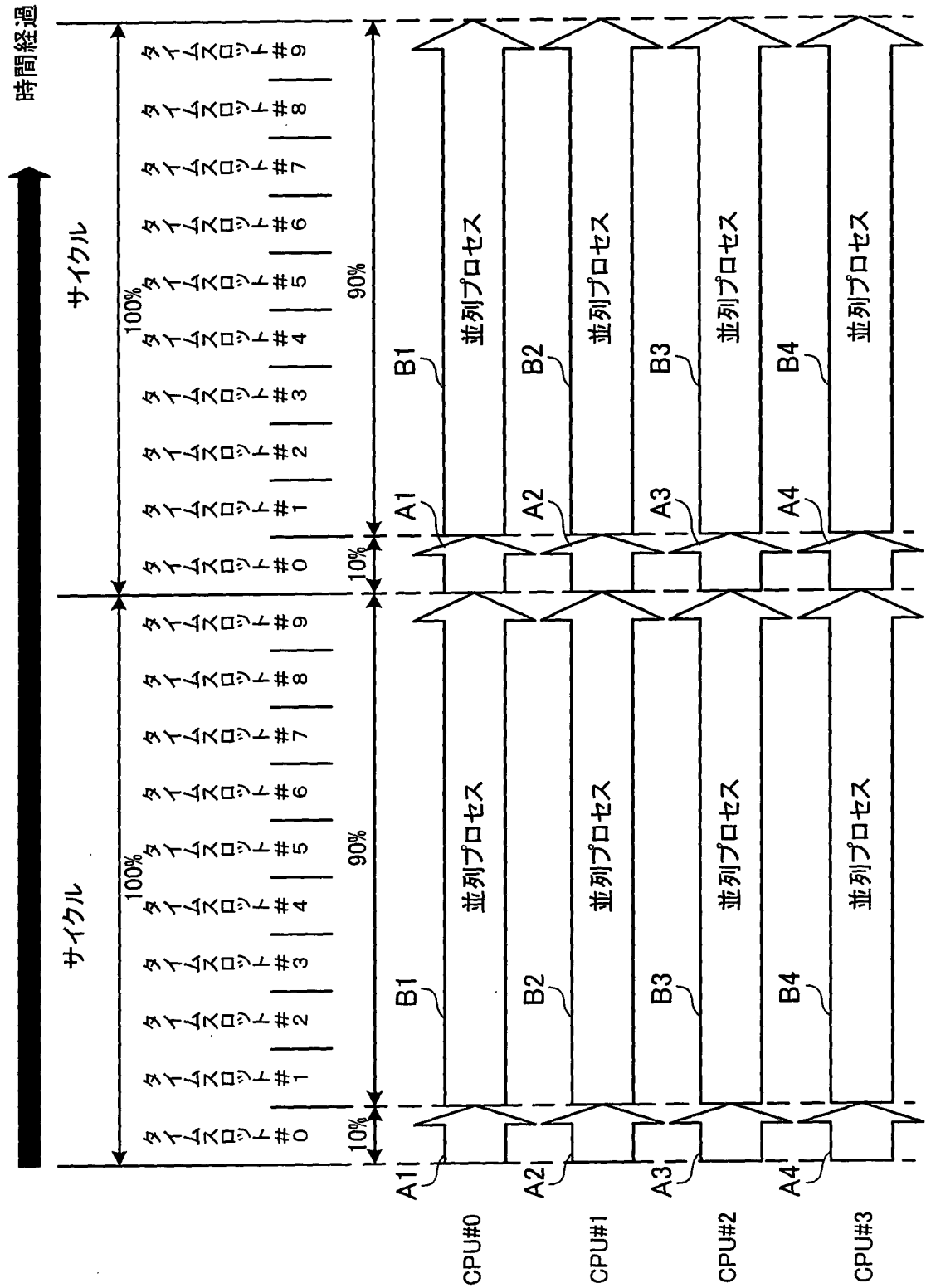
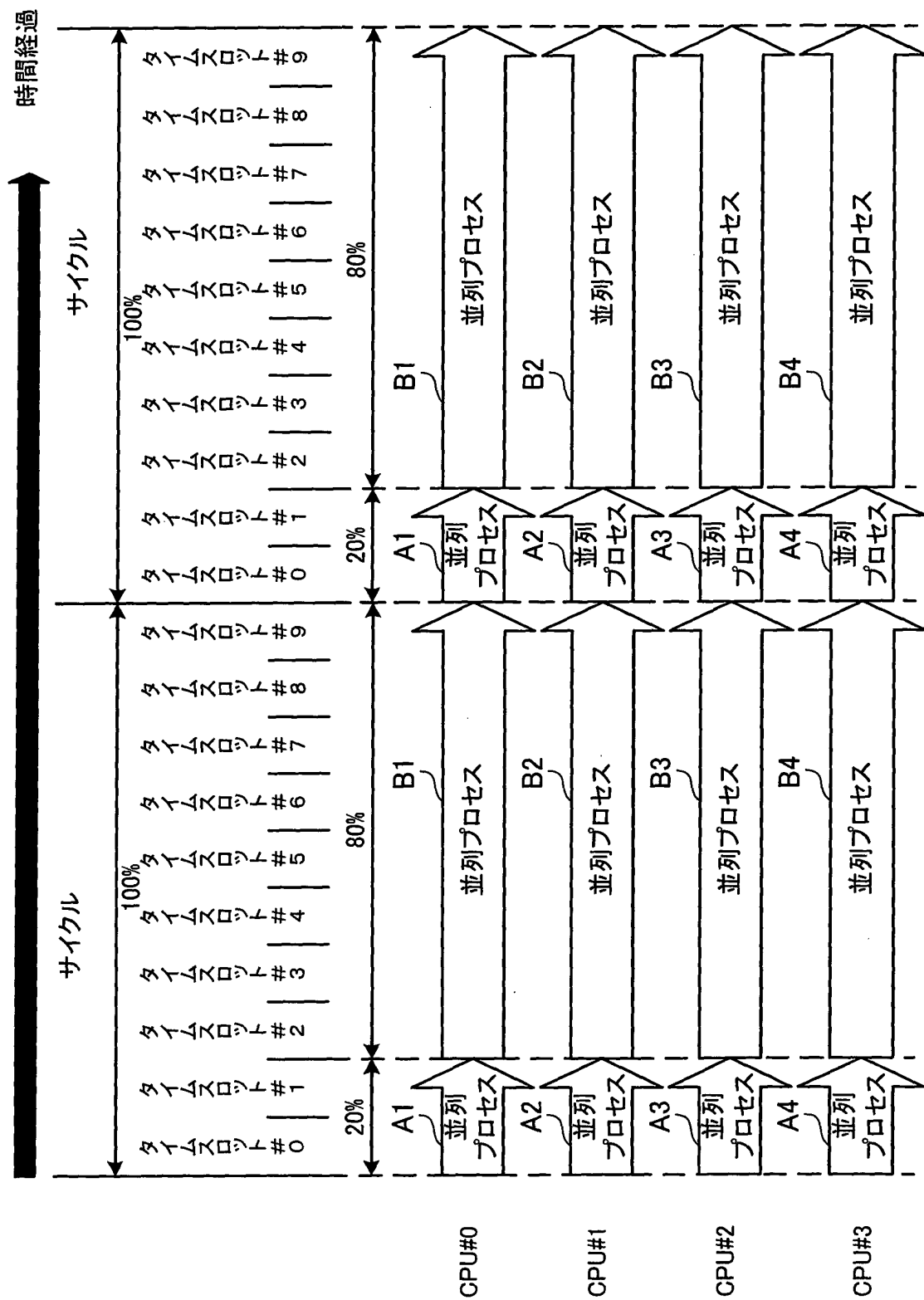
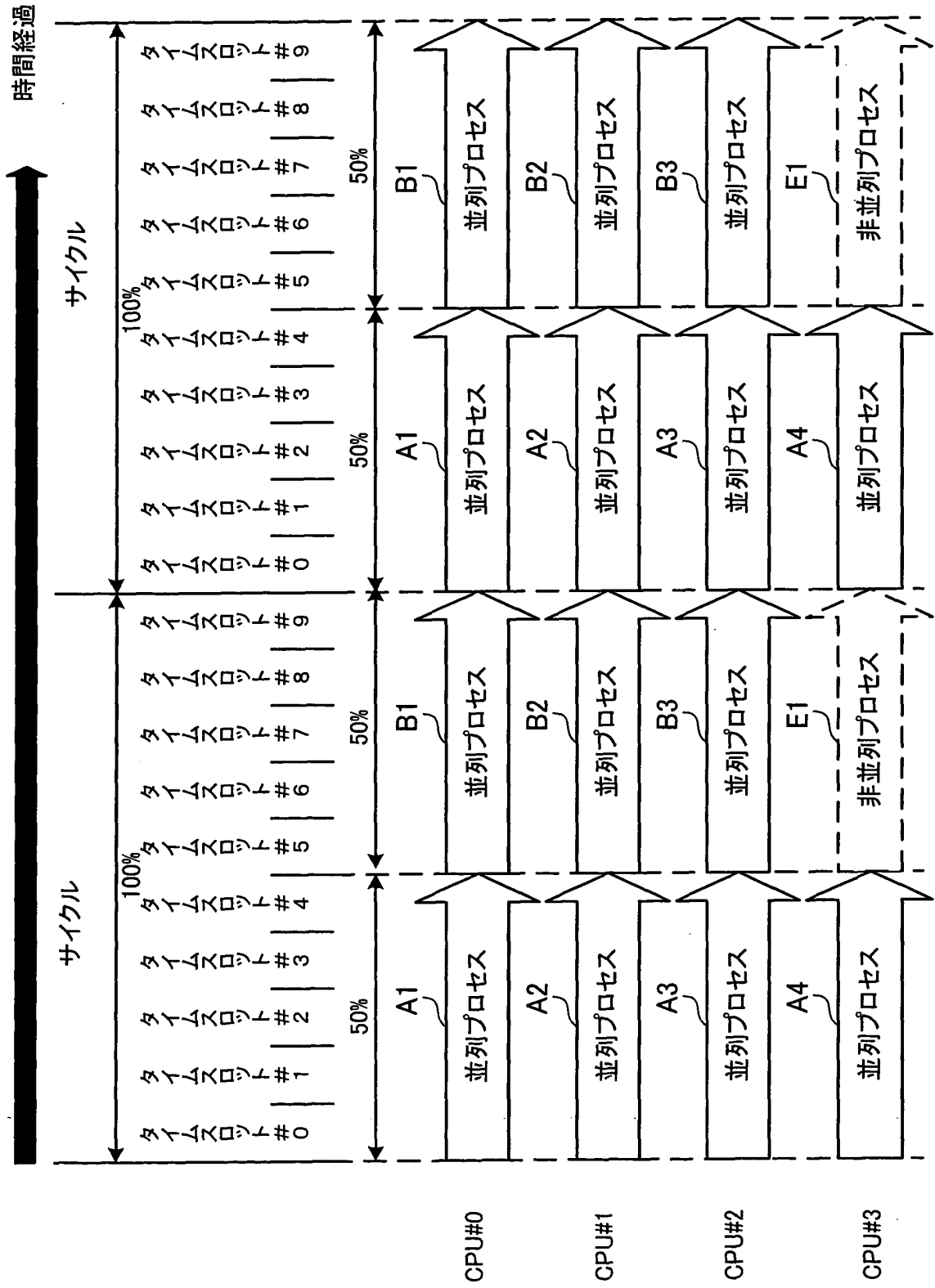
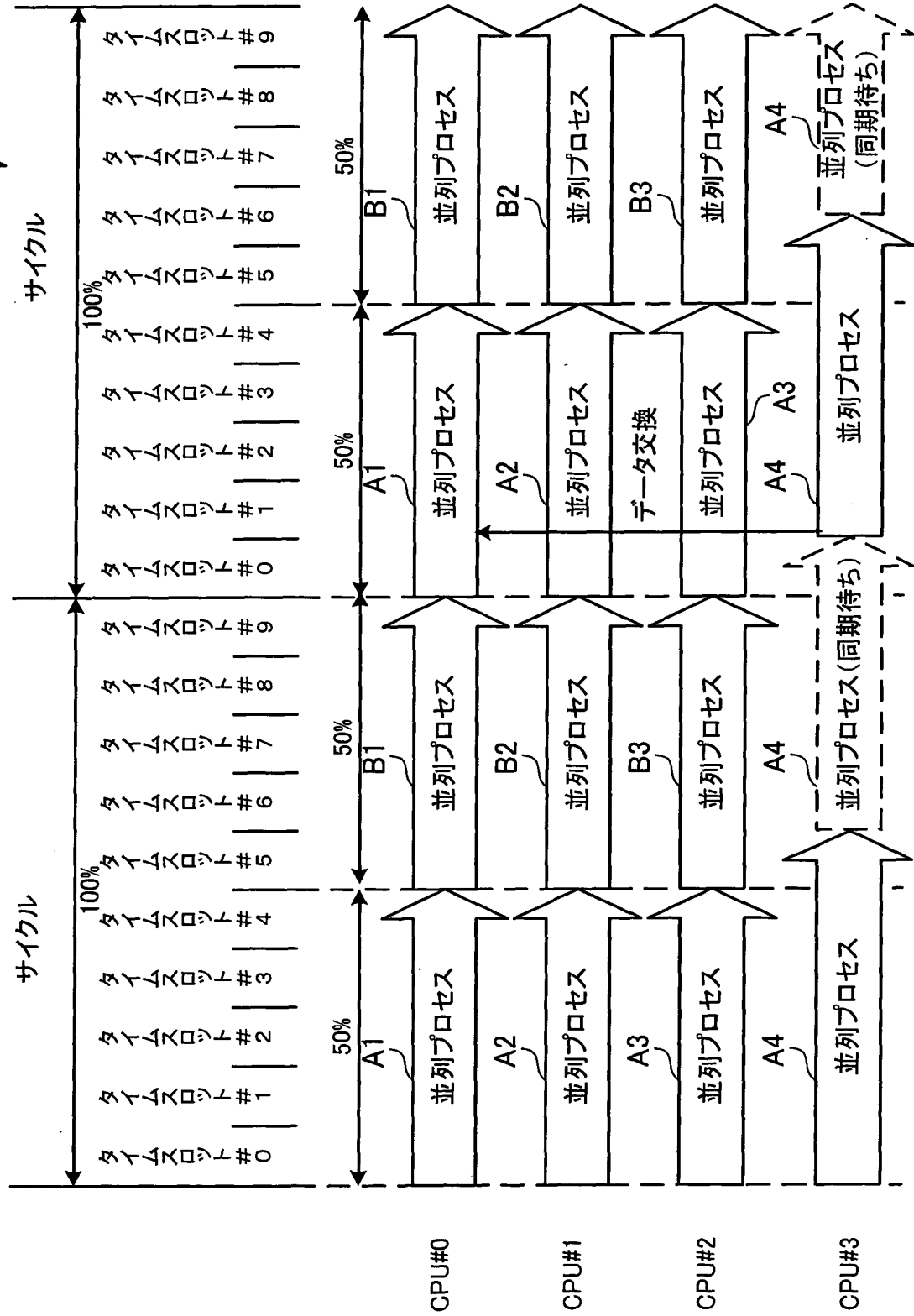


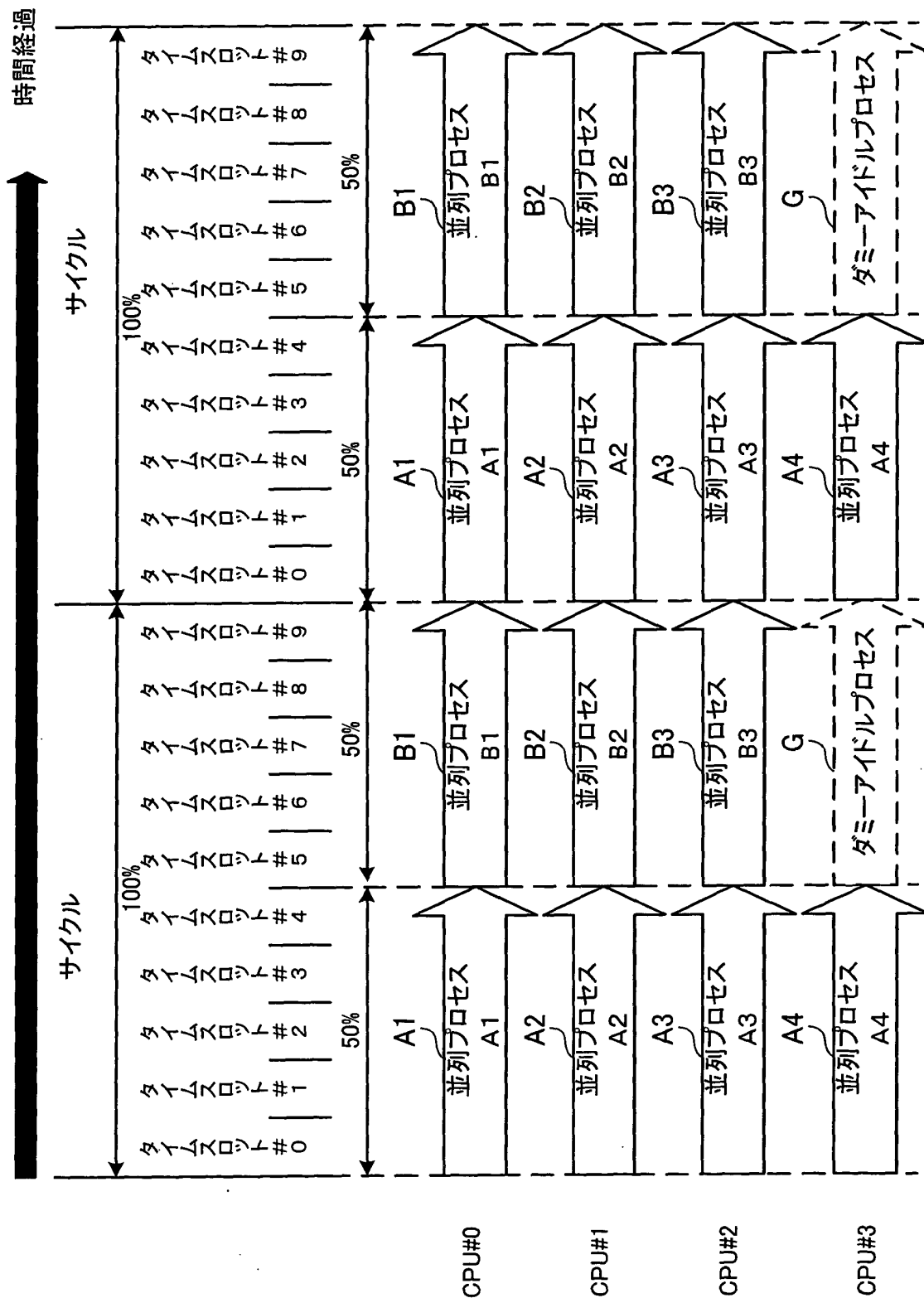
図12





時間經過





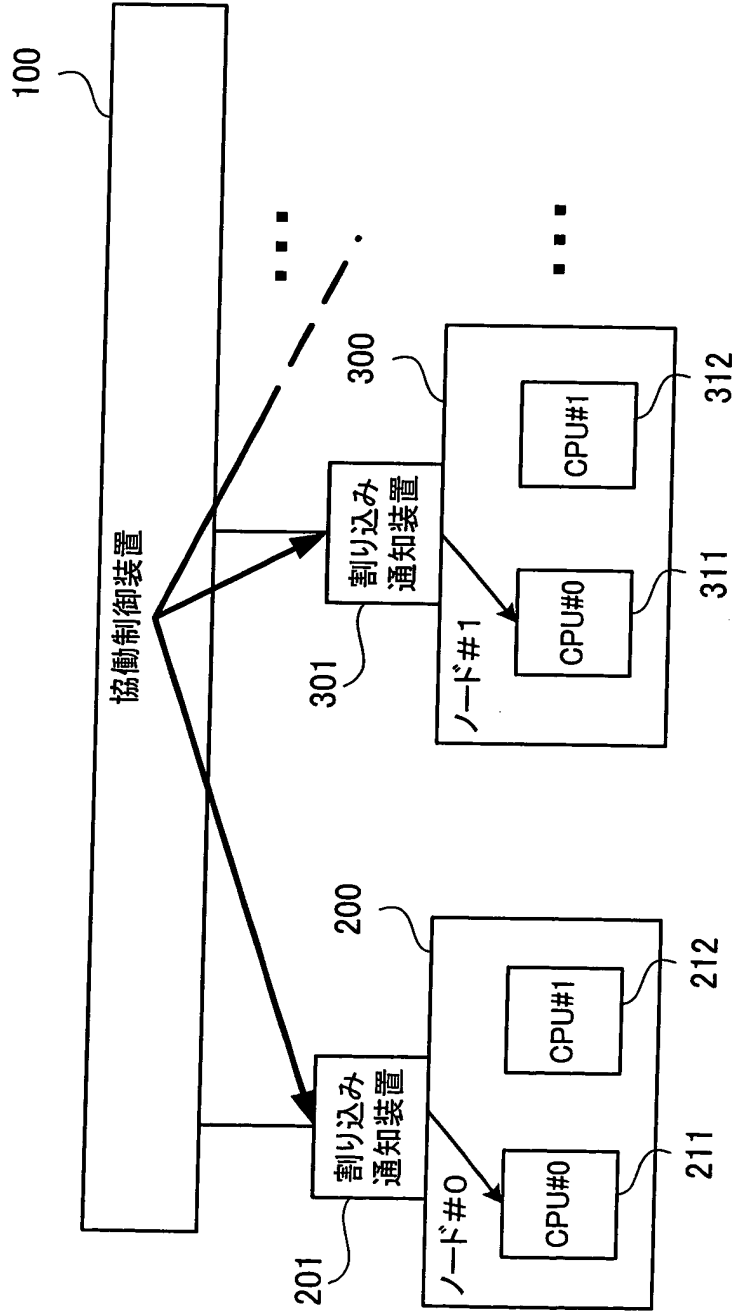


図16

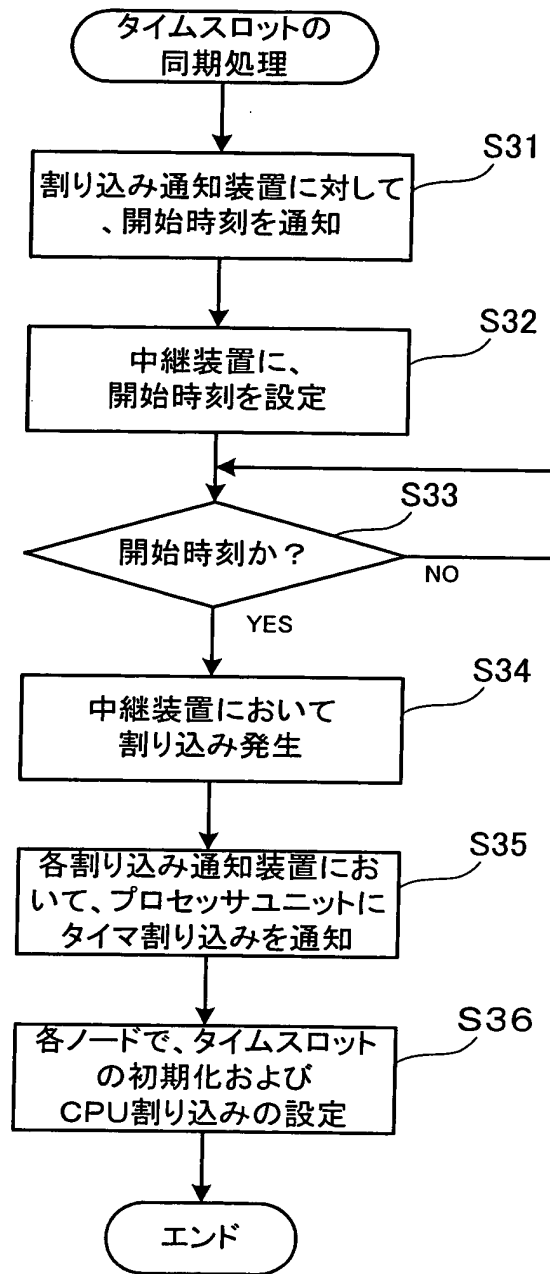


図17

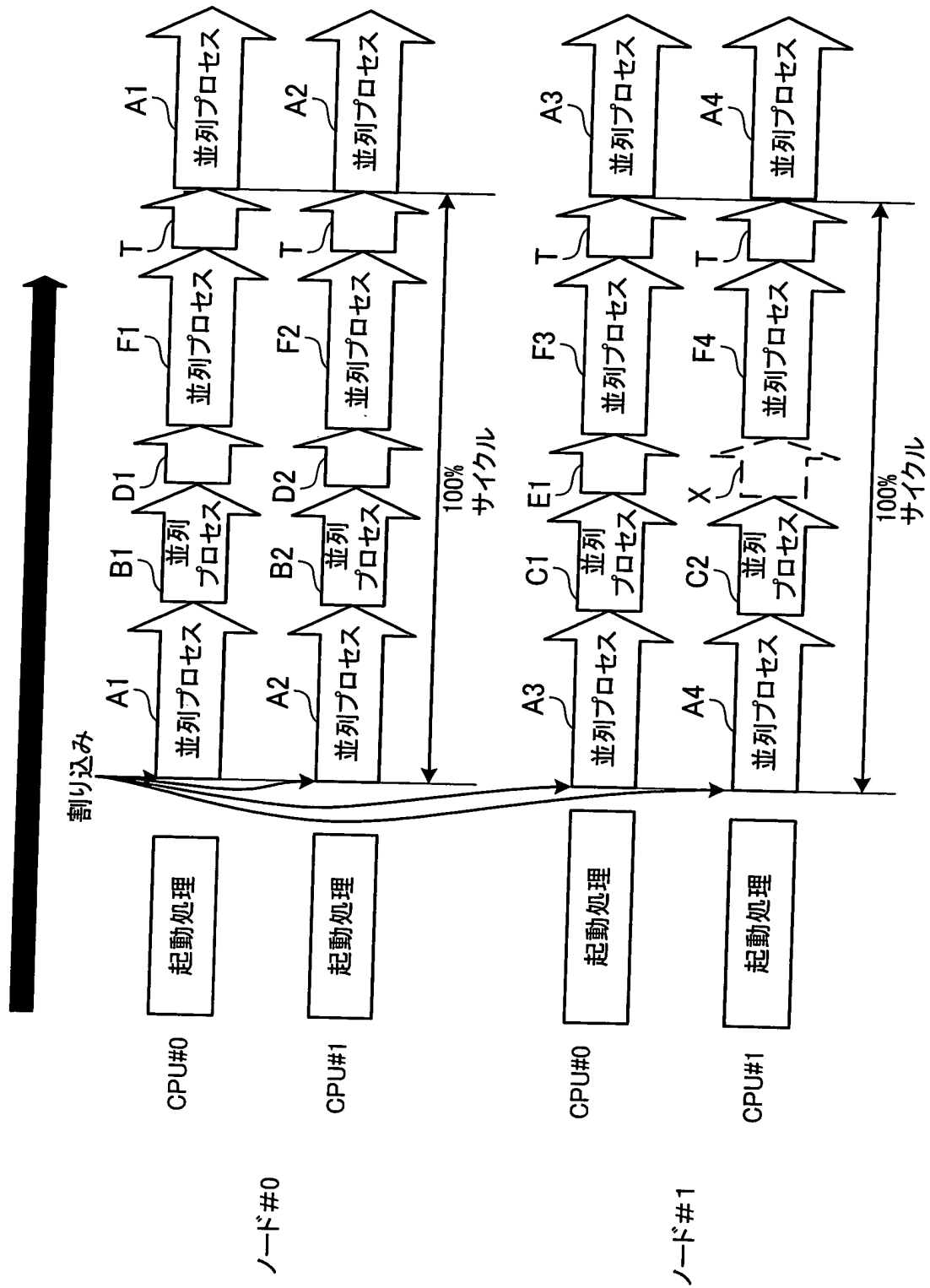


図18